Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут Електроенергетики
(інститут)
Електротехнічний факультет
(факультет)
Кафедра електропривода
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеню бакалавра
(бакалавра, спеціаліста, магістра)

студента Біди Павла Павловича
(ПІБ)
академічної групи 141-17-7
(шифр)
спеціальності 141 Електроенергетика, електротехніка та електромеханіка
(код і назва спеціальності)
спеціалізації[1] Електромеханічні системи автоматизації та електропривод

за освітньо-професійною програмою Електроенергетика, електротехніка та електромеханіка
(офіційна назва)
на тему Проєктування електропривода на базі синхронного двигуна із постійними магнітами із безконтактним датчиком температури ротора
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
Кваліфікаційної роботи	Балахонцев О.В.			
розділів:				
Спеціальна частина	Балахонцев О.В.			
Охорона праці	Столбченко О.В.			
Економічна частина	Тимошенко Л.В.			

Рецензент				

Нормоконтролер	Казачковський М.М.			

Дніпро
2021

ЗАТВЕРДЖЕНО:
завідувач кафедри
електропривода
(повна назва)

_____ Казачковський М.М.
(підпис) (прізвище, ініціали)

«_____»_____2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавра
(бакалавра, спеціаліста, магістра)

студенту Біда П.П. **академічної групи** 141-17-7
(прізвище та ініціали) (шифр)
спеціальності 141 Електроенергетика, електротехніка та електромеханіка

спеціалізації[1] Електромеханічні системи автоматизації та електропривод
за освітньо-професійною програмою Електроенергетика, електротехніка та електромеханіка
(офіційна назва)

на тему Проєктування електропривода на базі синхронного двигуна із постійними магнітами із безконтактним датчиком температури ротора**,** затверджену наказом ректора НТУ «Дніпровська політехніка» від 12.04.2021 № 201-с

Розділ	Зміст	Термін виконання
1. Terms of Reference	Опис проєкту, та актуальності проблеми виміру температури ротора.	01.04.2021-09.06.2021
2. Rotor Temperature Sensor	Опис принципу дії пристрою, розробка схем комунікацій та друкованої плати.	
2. Automated Electric Drive	Розробка блок-схеми, вибір електроприводів та додаткового обладнання. Вибір способів керування приводами. Розробка схем електричних підключень.	
3. Study of Dynamics of an Electric Drive	Розрахунок параметрів системи автоматичного регулювання і моделювання системи.	
4. Техніко-Економічне обґрунтування	Розрахунок капітальних та експлуатаційних витрат.	
5. Охорона праці	Аналіз небезпечних і шкідливих чинників, заходи до їх усунення. Розрахунок параметрів освітлення.	

Завдання видано _____ Балахонцев О.В.
(підпис керівника) (прізвище, ініціали)

Дата видачі 15 січня 2021

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Біда П.П.
(підпис студента) (прізвище, ініціали)

Реферат

Робота містить 122 аркуша, включаючи 19 рисунків та 14 посилань.

Темою дипломного проєктування є електропривод синхронного двигуна з постійними магнітами з вбудованим безконтактним датчиком температури ротора.

Мета роботи: розробка датчику температури, що вмонтовується на рухому частину двигуна (ротор) та транслює температурні дані із середини за протоколом бездротової передачі даних «Bluetooth».

У першому розділі розглянуто загальні відомості про синхронний двигун з постійними магнітами, актуальність і важливість виміру температури ротора.

У другому розділі виконано розробку безконтактного датчику температури.

У третьому розділі описані об’єкти автоматизованого електроприводу.

У четвертому розділі складена математична модель системи керування електропривода. Розраховані перехідні процеси системи електропривода.

В розділі охорони праці проведено аналіз небезпечних та шкідливих чинників, обрані заходи з охорони праці та виконано розрахунок освітлення для приміщення з об’єктом проєктування.

У розділі Техніко-Економічного Обґрунтування надано обґрунтування об’єкту проєктування, розраховані капітальні та експлуатаційні витрати.

Abstract

The work contains 122 pages, including 19 figures and 14 references.

The bachelor thesis topic is permanent magnet synchronous motor drive design with embedded wireless rotor temperature sensor.

Objective: embedded rotor temperature sensor design with wireless data transfer protocol “Bluetooth” implementation.

In the first chapter there is a common PMSM description, necessity of PMSM temperature measurement research.

In the second chapter rotor temperature sensor design process is described.

In the third chapter Automated Electric Drive objects are described.

In the fourth chapter Field Oriented Control system is described. Transient processes are simulated.

The section of labor protection analyzes hazardous and harmful factors, proposes measures for labor protection and calculates the lighting of the room where the design object is installed.

A separate section of the economic part provides a feasibility study of the design object, calculated capital and operating costs.

Table of Contents

1. Terms of Reference

					ЕП.ПД.21.201-С.ToR		
Зм.	Лист	№ докум.	Підпис	Дата			
Розробб.		Біда П.П.			PMSM Drive with Rotor Temperature Sensor	Лит.	Лист 7 / Листів 124
Перевірив		Балахонцев О.В.					
Н. Контр.		Казачковський М.М.				Павло Біда	
Затв.							

Project Description

In this project the Common Application Electric Drive based on PMSM with Wireless Rotor Temperature Sensor is designed. Major part of the project is dedicated to the Rotor Temperature sensor, what is made using existing ultra-low power technologies in RF data transmission, new concepts in power electronics and precise manufacturing abilities.

This project is made in *Esslingen University of Applied Sciences (Hochschule Esslingen)* for *MOTEG GmbH*. Some parts of this project are intellectual property of *MOTEG GmbH* and not included in this document.

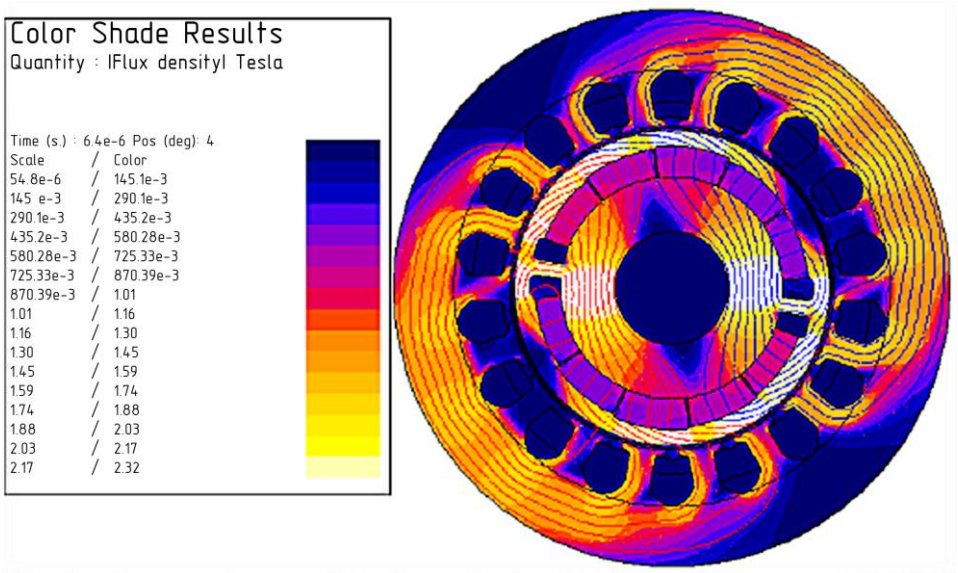
PMSM Rotor Temperature Estimation Necessity

As PMSM are usually more expensive than other motor types due to usage of expensive components, application of resource-saving methods is crucial. According to Report of Large Motor Reliability Survey of Industrial and Commercial Installations [1] one of the major causes of motor failure is overheating. Therefore, temperature control of the motor is important to prevent failures during the S1 and S3 operating modes.

It is important to mention, that there are no non-destructive techniques of temperature measurements inside the motor. Latest studies offer only predictive models, based on Kalman filter [2], Difference-Estimating Feedforward Neural Network [3] and High-Frequency Signal Injection [4].

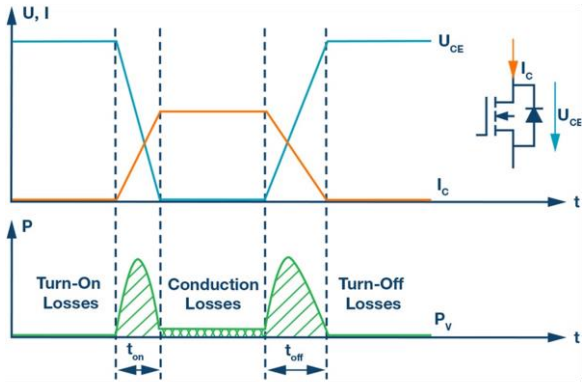
2. Rotor Temperature Sensor

Energy Harvesting is the Power Electronics concept. According to this concept, ultra-low power devices, such as modern MCU, can be powered from the ambient energy of the environment (Ex. Photovoltaic, Thermoelectric, Piezoelectric and RF signals). Despite the fact, that the concept is not new – realization of Energy Harvesting becomes possible only now.



1 PMSM Rotor Flux [5]

For powering the Rotor Temperature Sensor RF signals from the electromagnetic pole inside the motor are used, the powerful transistors of the frequency convertor create electromagnetic spikes during the transient process of commutating. In the beginning of the design process the hypothesis has been made. According to this hypothesis these spikes can be sensed in the magnetic pole of the PMSM and the power of the spikes can be “harvested” and used for the Rotor Temperature Sensor power.



2 IGBT Transistor Losses [6]

Spikes are occurring during the switch-on or switch-off transistor transient process.

We harvest this energy for powering the Rotor Temperature Sensor.

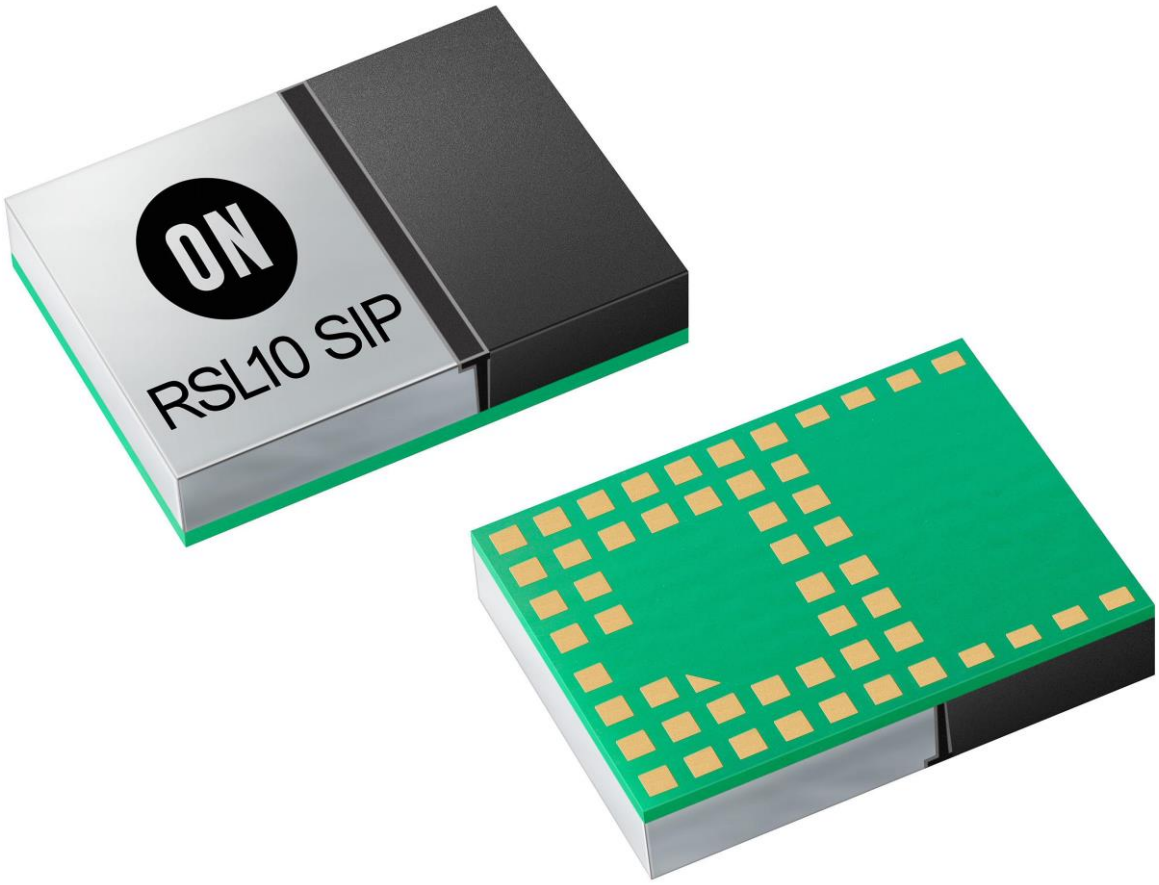
Conditions of spikes generation:

1. There is a current in drain-source of a transistor
2. There is a voltage drop in the transistor (transistor is not opened or closed)

The spikes frequency is dependent on the Frequency Converter's discretization value, transistor gate's capacitance and inductance

According to the assumptions, the next conditions must be fulfilled:

- All hardware has to be small, as it is supposed to be placed in a rotor slot
- Hardware has to be ultra-low power consumption class
- Hardware part has to be properly insulated from the rotor core
- Power of the Radio Transmitter has to be enough for breaking through motor



3 OnSemi RSL10 Chip [7]

The ON Semiconductor System on Chip RSL10 has been selected as the main MCU of the hardware part. It is low power System on Chip with Certified Bluetooth 5 on board, available both in WLCSP51 (BGA) and QFN48 packages.

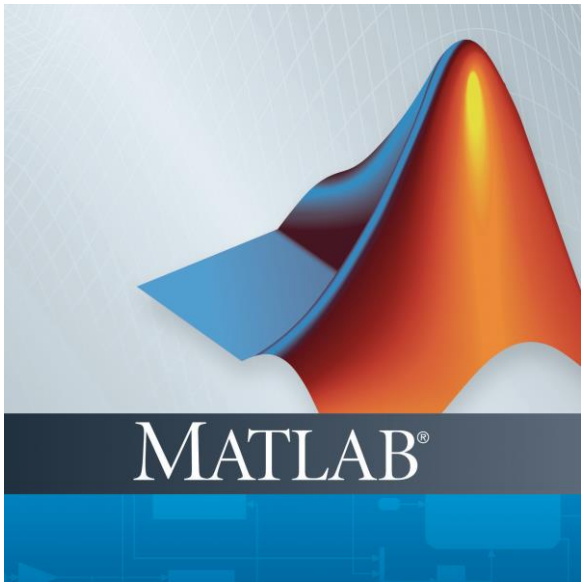
Despite nominal supply voltage is 3.63V and 3.3V as rated, during the tests it was estimated, that the lowest available voltage is 1.05V. At the lower levels MCU turns off. Turning on is possible with supply voltage level higher than 1.3V.

4 Bluetooth logo [8]

Bluetooth Low Energy (BLE) [8] is the low power version of Bluetooth that was built for the Internet of Things (IoT). This technology is widely used in the long period running applications where the main source of energy is coin-cell battery or energy-harvesting device.

Key benefits of the Bluetooth Low Energy:

- Strict Industry-standardized protocol
- Ultra-low power consumption
- Native support on various platforms
- Low-cost solution

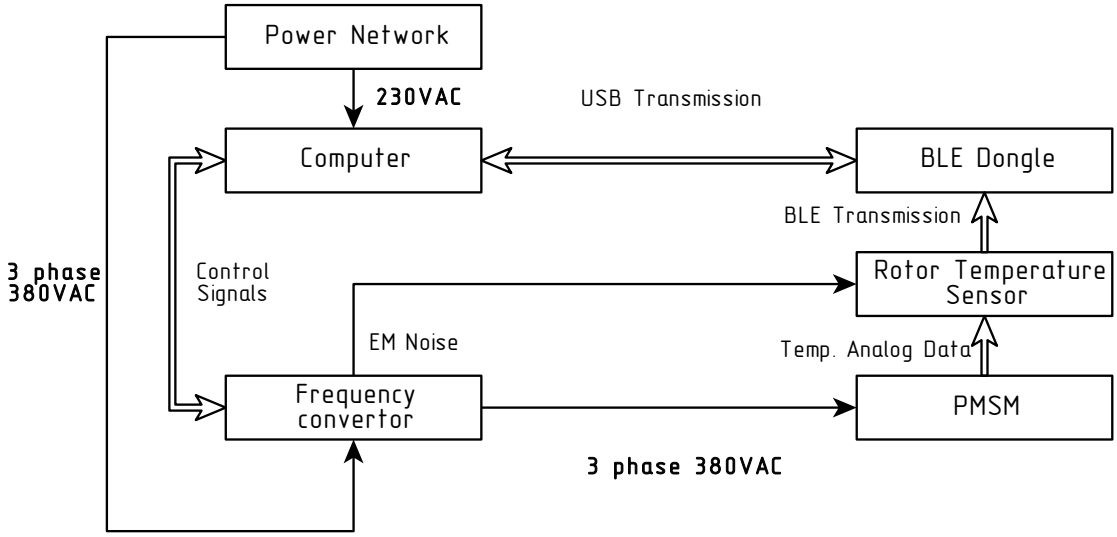


5 MATLAB logo [9]

MATLAB [9] is a computing platform for scientific research, data analysis, algorithm development and model creation. In this project MATLAB is used as front-end application for data receiving and plotting.

This tool has been selected due to reliability and easiness of use in applications like this project. Native support of BLE saved a lot of time during the R&D process.

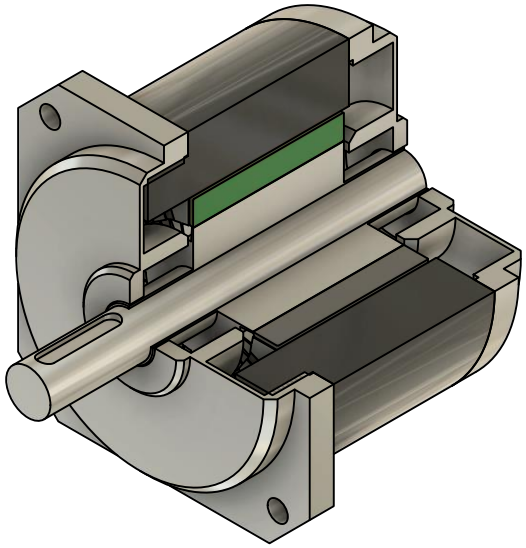
Assuming all previously selected technologies the scheme below shows how the whole assembled system will work.



6 Rotor Temperature Sensor System Scheme

Main elements description:

- Computer – the computing device, what runs MATLAB in OS Windows environment with an USB port. It takes data from USB port (virtual COM) and performs data processing within MATLAB environment. It communicates with Frequency Convertor via CAN bus as well. Powered from single-phase network.
- BLE Dongle – the Bluetooth Low Energy device (receiver). It is able to establish connection with the RSL10 MCU on the Rotor Temperature Sensor, receive wireless data and send it as Serial data to the USB port. Powered from Computer's USB port.
- Rotor Temperature Sensor – the device, powered from EM noise of Frequency Convertor inside the PMSM. Advertise itself to the ether, after establishing connection sends data from on-board temperature sensors periodically.

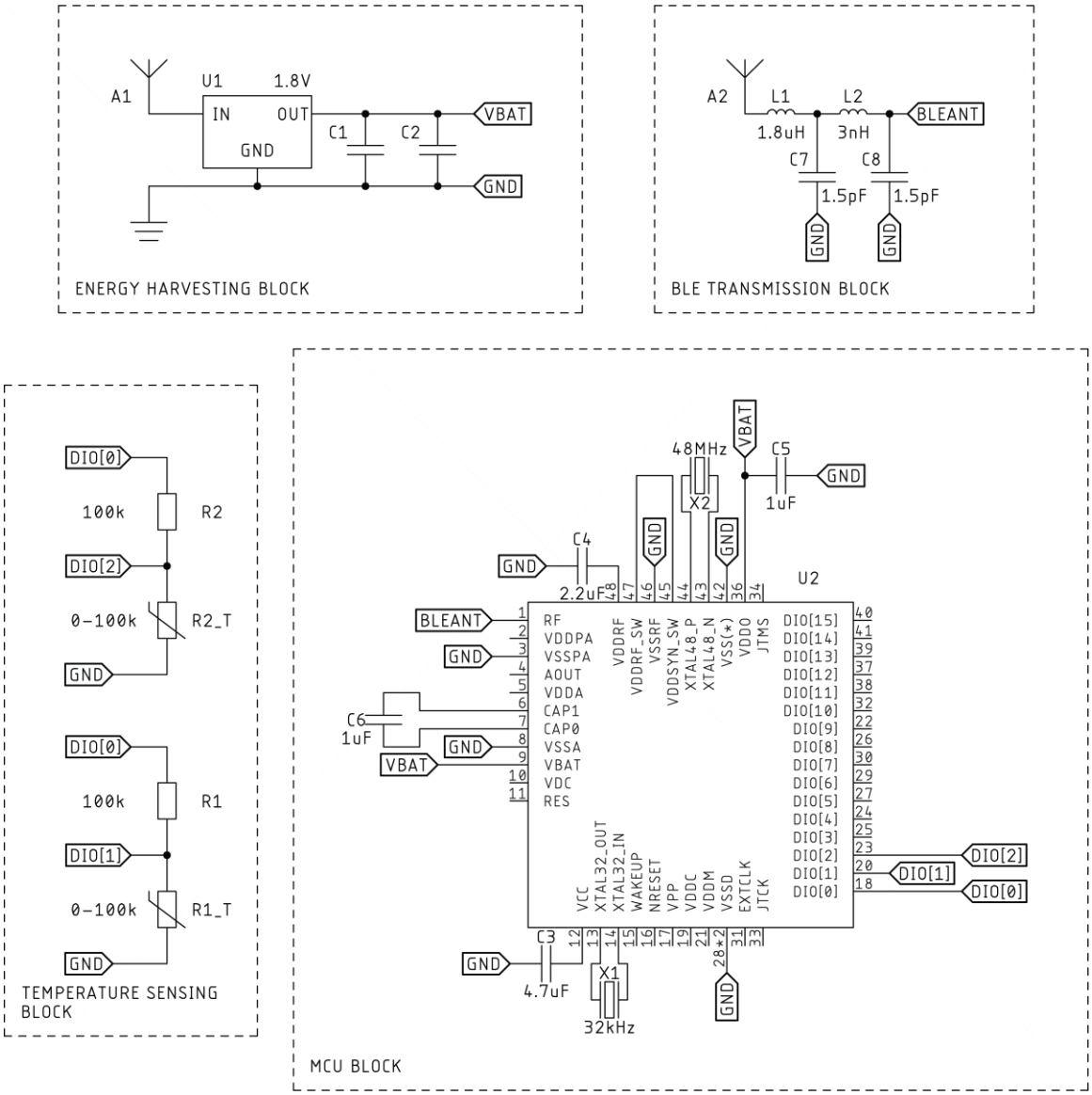


7 PMSM Motor with Rotor Temp. Sensor

Another important task in the project – the decision of the Rotor Temperature Sensor displacement. The best placement of the device is the motor's rotor slot.

For the electromagnetic and mechanical balance of the rotor two Rotor Temperature Sensors are placed in the opposite rotor slots, this solution helps to filter noise in received temperature data.

Hardware Part



8 Rotor Temperature Sensor Schematic

The schematic diagram consists of 4 blocks: Energy Harvesting, BLE Transmission, Temperature Sensing and MCU Block.

All these blocks are located on the same Printed Circuit Board. The variant of splitting the design between two circuit boards (Energy Harvesting and MCU) and locating them inside opposite rotor slots, for the better rotor balance has been considered, but for now the whole design (except the energy harvesting antenna) is located on the same PCB.

Energy Harvesting Block downsizes the voltage, inducted onto Antenna (A1) to the level of 1.8V with help of Linear Stabilizer (U1). Capacitors (C1, C2) required for the voltage stabilization. Stabilized power can be taken between VBAT and GND outputs.

BLE Transmission Block is a typical Bluetooth transceiver scheme. It takes 2.4GHz signal at BLEANT input and transmit it into the ether. Antenna (A2) is placed directly onto a PCB as a copper trace.

Temperature Sensing Block consists of 2 voltage dividers. Both of the voltage dividers are powered from the MCU pin (DIO[0]) only during the measurements in behalf of energy consumption decrease. Each voltage divider consists of 100k resistor (R1, R2) and PTC thermistor (R31, R33). Voltage drop across thermistors is measured by 14bit ADC of MCU. Voltage range is [0V:0.9V]. The temperature dependence from the voltage drop across thermistor is not linear.

Calculation of the core voltage:

1. Reading ADC_0 value from the DIO[0]
2. Calculating the voltage value $V_{BATT} = \frac{ADC_0}{8192}$ [V]

For the calculation of the temperature value the next steps has to be taken:

1. Reading ADC_n value from the DIO[1] or DIO[3]
2. Calculating the voltage value $V_n = \frac{ADC_n}{8192}$ [V]
3. Calculating the current in the voltage divider $I_n = \frac{V_{BATT} - V_n}{R_x}$ [A], where R_x is R1 or R2 value
4. Calculating the thermistor resistance $R_n = \frac{V_n}{I_n}$ [Ohms]
5. Getting the temperature data from the function $f(x) = T(R)$, defined in the thermistor datasheet

MCU Block consists of RSL10 MCU itself and required periphery as filtering capacitors and quartz resonators.

Firmware Part

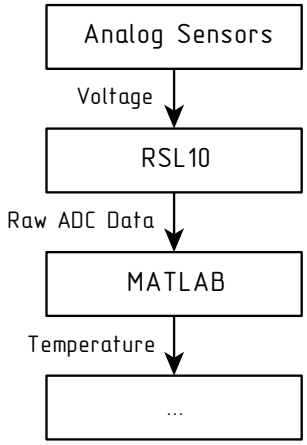
Adress	[20:12]	[11:8]	[7:4]	[3:0]
Value	-	U_{R33}	U_{R31}	$\frac{U_{BATT}}{2}$
Source		DIO[3]	DIO[1]	Internal

Data Packet Structure

Gathered data is sent to the transceiver in data packets with frequency of 6 Hz (it is not crucial due to high temperature inertia in the motor). Each data packet consists of 3 values:

1. Input voltage value divided by two (internal register value)
2. Voltage drop across R_{31}
3. Voltage drop across R_{33}

All the values are transmitted as Raw ADC input in range of [0:16384]



9 Rotor Temperature Sensor Order of Calculation

From the analog sensors, such as voltage thermistor dividers and internal voltage inputs of RSL10 data is gathered and transmitted to the Receiver device (USB Dongle). The Receiver transmits data to the MATLAB via Serial port (Emulated RS-232). MATLAB converts ADC input to Voltage drop, then Voltage drop to active Resistance of the thermistor and then (using the Resistance/Temperature lookup tables) converts Resistance values to the Temperature data.

For the creation of firmware free Integrated Development Environment “On Semiconductor IDE” has been used. This IDE has native support of RSL10 and provides some code examples. The firmware source code is mostly inherited from the *ble_peripheral_server* [7] example. All the firmware source code is written in C language.

Source code is listed in *Appendix A* section.

Software Part

This part is about the software, located on the Front-End side of the system. In this project the Front-End side is located and executed with MATLAB application in the environment of OS Windows.

The hardware and firmware parts create next requirements for the software:

- Ability to control the Serial ports (COM ports)
- Ability to control the embedded Bluetooth
- Ability to process, analyze and plot received data

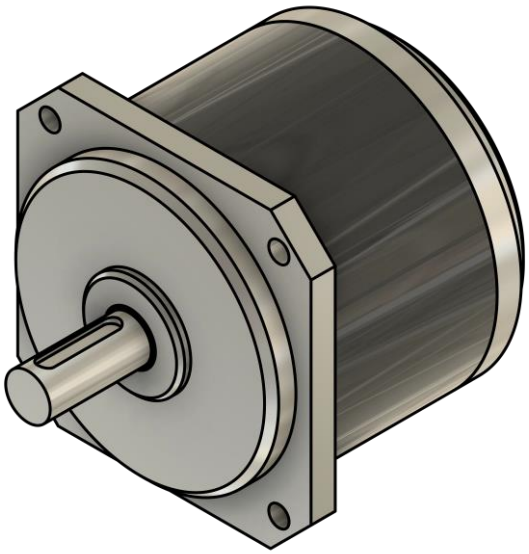
All the requirements can be fulfilled with the abilities of pure MATLAB 2020b without toolboxes.

Script Name	Description
RTS.m	The main software script all other scripts are called from here.
RTS_INIT.m	Initialize the BLE connection, handles the errors and exceptions. After the successful connection initialize the Characteristic with further ability of reading it.
RTS_LOOKUP.m	Contains the Temperature/Resistance Lookup Tables of the thermistors in the array of points. These points are interpolated for getting the required temperature value.
RTS_READ	Requests the data from the Rotor Temperature Sensor, and receives it, adds the time spent on the data transfer to the common time counter. Returns the values of the battery voltage and thermistors' voltage drops.

3. Automated Electric Drive

					ЕП.ПД.21.201-С.AED			
Зм.	Лист	№ докум.	Підпис	Дата				
Розробб.		Біда П.П.			PMSM Drive with Rotor Temperature Sensor	Лит.	Лист	Листів
Перевірив		Балахонцев О.В.					24	124
						Павло Біда		
Н. Контр.		Казачковський М.М.						
Затв.								

Electric Motor Description



10 MOTEG PMSM Motor

Permanent Magnet Synchronous Motors (PMSM) are widely used for the industrial applications. Mechanisms without speed control requirement are often driven by PMSMs (Ex. Compressors, Pumps, Fans, Generators).

In the applications where speed control is required both scalar and vector control can be used. Despite it is more expensive, vector control is used widely, as it has set of advantages like precision and smoothness of speed.

As the current project is performed for the MOTEG GmbH, the PMSM in this project is taken from the MOTEG PMSMs catalogue.

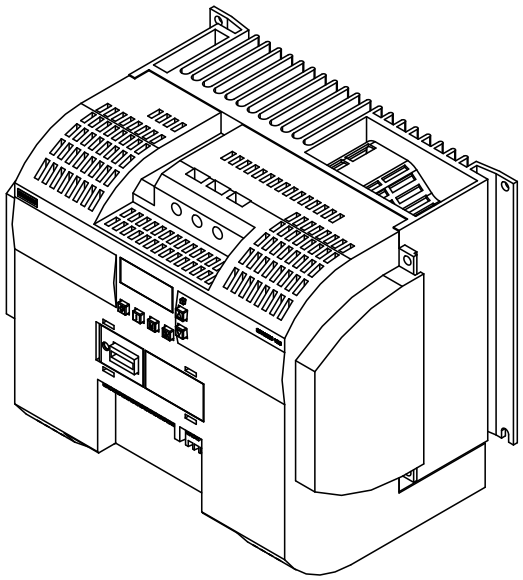
Motor Rated Parameters [10]:

Name	MOTEG MMA 80-8-60-C-…-W
Power on the Shaft	5.6 kW

Nominal Torque	18 Nm
Max. Torque	21.5 Nm
Nominal Speed	3000 rpm
Max. Speed	4000 rpm
Nominal Current	9.5 A
Max. Current	11.5 A
Weight	9.5 kg
Protection Class	IP67

This Motor is driven with FOC (field orientated control). Siemens Frequency Convertor is used for control.

Frequency Convertor



The frequency converter is designed to transfer alternating current from one frequency to another, that is, to change its voltage, rectifying the mains and converting it to three-phase.

11 SINAMICS V20 6SL3210 [11]

To choose and buy a frequency converter correctly, you need to take into account some factors:

- The output currents of the inverter must be less than the full load current;
- overload torque, for example, if you install an inverter with a special harmonic filter, then the supply transformer will be loaded less.

AC mains frequency converters can be:

- electronic type and work in generator mode;
- made in one case and in two;
- have all kinds of rectifier configurations (6, 12, 18 pulse);

- there are several types of cooling (liquid, air and back wall);
- type of load (with continuous moment, inertial, inert, active, crane);
- various protection class of the case (IP00, IP20, IP21, IP55, IP66).

The automatic form of the inverter is not complicated and less perfect. The electronic type of the converter creates at the output an electric voltage of a given frequency and is used in electric motors of modulating frequency control, providing its absolute protection against short circuits.

The converter in electric motors guarantees:

- smooth start-up;
- smooth stop;
- effective protection against overloads and voltage surges;
- works with high rated mains voltages and currents;
- endures maximum operating frequency, impulse actions and continuous load;
- by adjusting technological parameters, power consumption is reduced and energy saving is increased;
- improves energy performance;

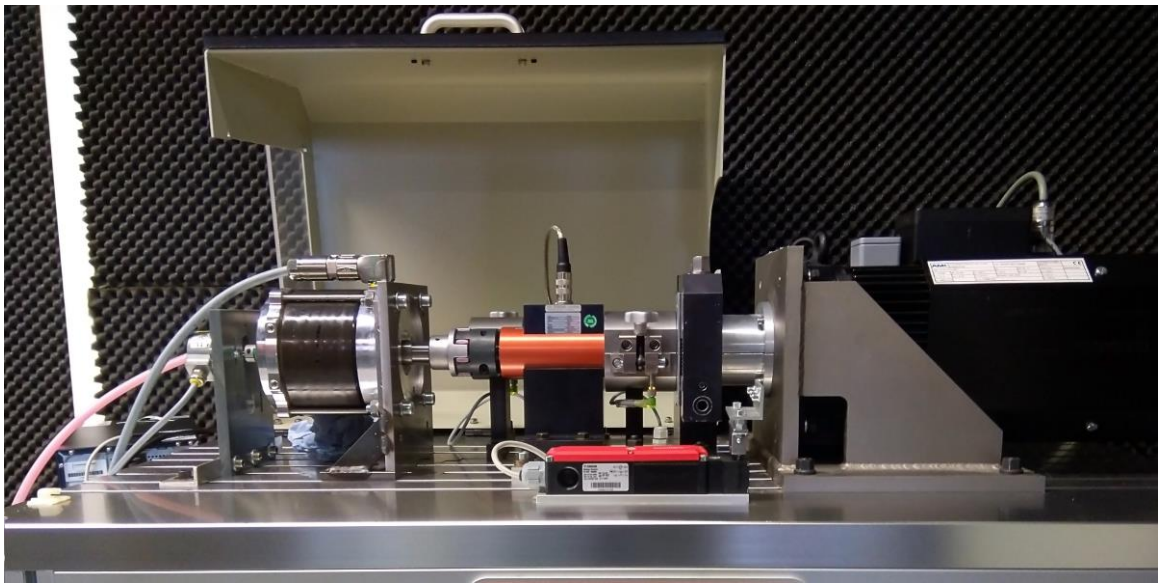
- softens mechanical characteristics;
- aligns the technological process and makes it more accurate and correct;
- increases the period of operation of technological equipment;
- increases efficiency and production prosperity (reduces downtime);
- facilitates the work and execution of difficult automatic tasks.

Siemens SINAMICS V20 6SL3210-5BE31-1CV0 has been selected for the purpose of driving the PMSM in this project.

4. Study of Dynamics of an Electric Drive

					ЕП.ПД.21.201-С.StDED			
Зм.	Лист	№ докум.	Підпис	Дата				
Розробб.		Біда П.П.			PMSM Drive with Rotor Temperature Sensor	Лит.	Лист	Листів
Перевірив		Балахонцев О.В.					31	124
						Павло Біда		
Н. Контр.		Казачковський М.М.						
Затв.								

Description of the model



12 Test Workbench

For the current setup next differential equations system of EMF is valid:

$$\begin{cases} u_a = R_1 \cdot i_a + \frac{d\psi_a}{dt}, \\ u_b = R_1 \cdot i_b + \frac{d\psi_b}{dt}, \\ u_c = R_1 \cdot i_c + \frac{d\psi_c}{dt}. \end{cases}$$

Flux linkage in phases:

$$\psi_a = L_a \cdot i_a + L_{ab} \cdot i_b + L_{ac} \cdot i_c + \sum_{\mu} \Psi_{2m\mu} \cdot cos \ \ \mu \cdot \theta \,,$$

$$\psi_b = L_b \cdot i_b + L_{ab} \cdot i_a + L_{bc} \cdot i_c + \sum_{\mu} \Psi_{2m\mu} \cdot cos(\mu \cdot \theta - 2 \cdot \pi/3) \,,$$

$$\psi_c = L_c \cdot i_c + L_{ac} \cdot i_a + L_{bc} \cdot i_b + \sum_{\mu} \Psi_{2m\mu} \cdot cos(\mu \cdot \theta + 2 \cdot \pi/3),$$

where $L_a = L_b = L_c = L_w$ – main phase inductance;

$L_{ab} = L_{ac} = L_{bc} = -L_w/2$ – main mutual inductance between stator phases;

$\sum_{\mu} \Psi_{2m\mu} \cdot cos \ \ \mu \cdot \theta$ – rotor flux linkage with stator winding;

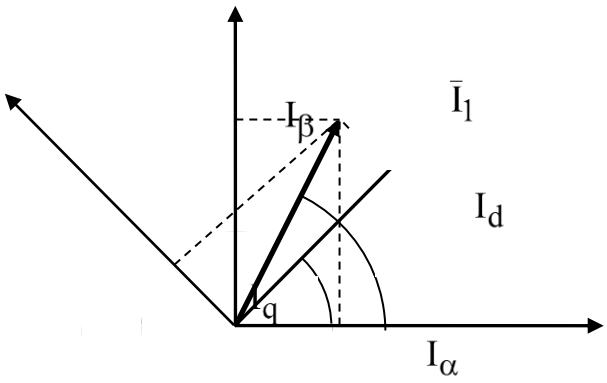
$\mu = 6 \cdot k + 1, \quad k = 1, \ \pm 2, \ \pm 3, \ \ldots$ – index of flux linkage harmonic;

θ – rotor electrical angle.

Well known space vectors method is used for simplicity. For any of three instantaneous voltage, current and flux linkage, there is a single generalized space vector, rotating inside the air gap with the grid frequency:

$$\bar{I}_1 = \frac{2}{3} \cdot (i_a + i_b \cdot \bar{a} + i_c \cdot \bar{a}^2)$$

where $\bar{a} = e^{j2 \cdot \pi/3}$ – vector with consideration of the space winding shift.



13 Connection between static and moving axis

As space vectors are rotating into the complex plane, they can be projected on the axis. But stator variables are visualized into the static coordinate system $\alpha,\ \beta$, where real axis α is aligned to A. For transition between instantaneous values of different phases to the static coordinate system $\alpha,\ \beta$ Clarke transformations are used:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}.$$

Rotor variables are shown in the static coordinate system $d,\ q$, which is synchronous to the rotor, where real axis d is colinear with the pole axis. For transition to the moving coordinate system Park transformation is used:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \begin{bmatrix} cos\,\theta & sin\,\theta \\ -sin\,\theta & cos\,\theta \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix}.$$

Both transformations can be combined as:

$$\begin{bmatrix} i_d \\ i_q \end{bmatrix} = \frac{2}{3} \cdot \begin{bmatrix} cos\,\theta & cos(\theta - 2\pi/3) & cos(\theta + 2\pi/3) \\ -sin\,\theta & -sin(\theta - 2\pi/3) & -sin(\theta + 2\pi/3) \end{bmatrix} \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}.$$

Sometimes it is necessary to perform an inverse transformation it can be achieved the next way:

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1/2 & \sqrt{3}/2 \\ -1/2 & -\sqrt{3}/2 \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix},$$

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} cos\,\theta & -sin\,\theta \\ sin\,\theta & cos\,\theta \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix},$$

$$\begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} cos\,\theta & -sin\,\theta \\ cos(\theta - 2\pi/3) & -sin(\theta - 2\pi/3) \\ cos(\theta + 2\pi/3) & -sin(\theta + 2\pi/3) \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix}.$$

Thus the system of equations can be represented as space vectors in the static coordinate system $\alpha,\ \beta$:

$$\bar{U}_{1\alpha\beta} = R_1 \cdot \bar{I}_{1\alpha\beta} + \frac{d\bar{\Psi}_{1\alpha\beta}}{dt},$$

where $\bar{\Psi}_{1\alpha\beta} = L_1 \cdot \bar{I}_{1\alpha\beta} + \bar{\Psi}_{2\alpha\beta}$ – flux linkage of the resulting field with stator's winding; L_1 – stator full inductance, $\bar{\Psi}_{2\alpha\beta}$ – rotor field flux linkage.

Flux linkage space vector of the resulting field:

$$\bar{\Psi}_{1dq} = \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix} \cdot \begin{bmatrix} i_d \\ i_q \end{bmatrix} + \bar{\Psi}_{2dq}$$

where $L_d = L_{ad} + L_\sigma,\ L_q = L_{aq} + L_\sigma$ – main inductances of colinear and perpendicular axes; L_σ – leakage inductance;

$$\bar{\Psi}_{2dq} = \begin{bmatrix} \Psi_1 + (\Psi_5 + \Psi_7) \cdot cos & 6 \cdot \theta + (\Psi_{11} + \Psi_{13}) \cdot cos & 12 \cdot \theta + \cdots \\ (-\Psi_5 + \Psi_7) \cdot sin & 6 \cdot \theta + (-\Psi_{11} + \Psi_{13}) \cdot sin & 12 \cdot \theta + \cdots \end{bmatrix}$$

Flux linkage of the rotor field with consideration of high harmonics. Non-sinusoidal distribution of the magnetic field causes dependance of main inductances to the rotor position. For the modeling simplification we will consider only the dependance of the main magnetic field harmonic.

It is enough to consider only first four flux linkage harmonics as higher ones have insufficient effect on the torque pulsation. For rotor field flux linkage vector projection:.

$$\psi_{2dq} = \begin{bmatrix} \Psi_1 + \psi_{d6} \cdot cos \ \ 6 \cdot \theta + \psi_{d12} \cdot cos \ \ 12 \cdot \theta \\ \psi_{q6} \cdot sin\, 6 \cdot \theta + \psi_{q12} \cdot sin \quad 12 \cdot \theta \end{bmatrix}$$

Where $\psi_{d6},\ \psi_{q6},\ \psi_{d12},\ \psi_{q12}$ – harmonics amplitudes of corresponding rotor field flux linkage components.

Differential equation of the stator voltage in the moving coordinate system $d,\ q$:

$$\bar{U}_{1dq} = R_1 \cdot \bar{I}_{1dq} + \frac{d\bar{\Psi}_{1dq}}{dt} + j \cdot \omega_e \cdot \bar{\Psi}_{1dq}$$

where ω_e – electric rotor angular speed.

$$u_d = R_1 \cdot i_d + L_d \cdot \frac{di_d}{dt} - \omega_e \cdot (L_q \cdot i_q - (-\psi_{q6} - 6 \cdot \psi_{d6}) \cdot sin \ \ 6 \cdot \theta +$$

$$-(-\psi_{q12} - 12 \cdot \psi_{d12}) \cdot sin \ \ 12 \cdot \theta),$$

$$u_q = R_1 \cdot i_q + L_q \cdot \frac{di_q}{dt} + \omega_e \cdot (L_d \cdot i_d + \Psi_1 +$$

$$+(\psi_{d6} + 6 \cdot \psi_{q6}) \cdot cos \ \ 6 \cdot \theta + (\psi_{d12} + 12 \cdot \psi_{q12}) \cdot cos \ \ 12 \cdot \theta).$$

Amplitudes of rotor field flux linkage in moving coordinate system:

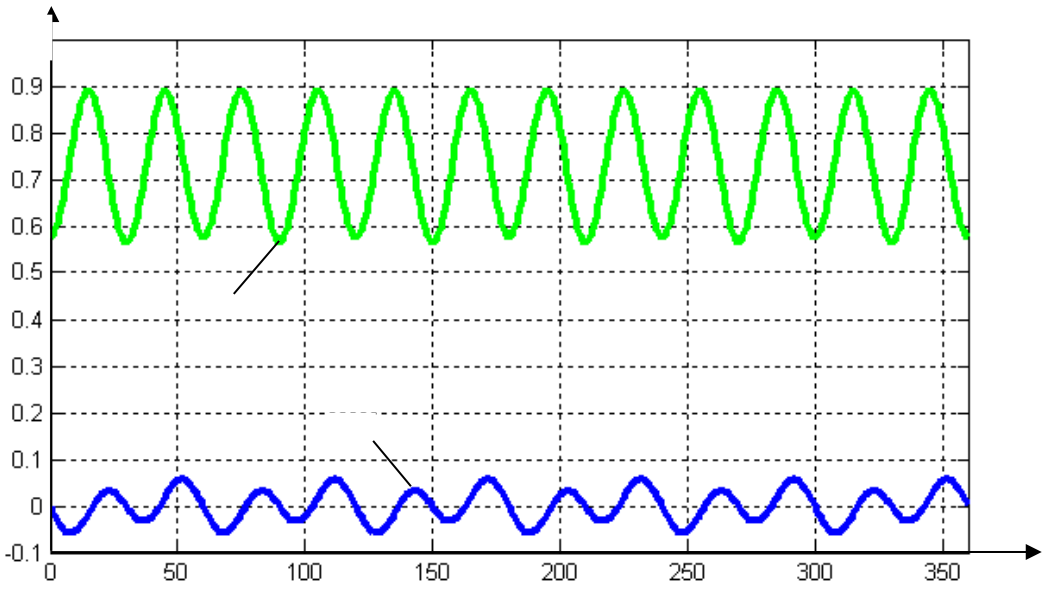
$$\psi_{d6} = -\psi_{q6} - 6 \cdot \psi_{d6} = -(-\Psi_5 + \Psi_7) - 6 \cdot (\Psi_5 + \Psi_7)$$

$$= -5 \cdot \Psi_5 - 7 \cdot \Psi_7$$

$$\psi_{q6} = \psi_{d6} + 6 \cdot \psi_{q6} = (\Psi_5 + \Psi_7) + 6 \cdot (-\Psi_5 + \Psi_7) = -5 \cdot \Psi_5 + 7 \cdot \Psi_7$$

$$\bar{\psi}_{2dq} = \begin{bmatrix} \Psi_d(\theta) \\ \Psi_q(\theta) \end{bmatrix} = \begin{bmatrix} \Psi_{d6} \cdot sin\, 6 \cdot \theta + \Psi_{d12} \cdot sin \ \ 12 \cdot \theta \\ \Psi_1 + \Psi_{q6} \cdot cos \ \ 6 \cdot \theta + \Psi_{q12} \cdot cos \ \ 12 \cdot \theta \end{bmatrix}$$

Rotor field flux linkage dependance on the rotor angle in the moving coordinate system:



14 Space vector parts of the rotor field flux linkage in the system

After slight transformation of the equations:

$$u_d = R_1 \cdot i_d + L_d \cdot \frac{di_d}{dt} - \omega_e \cdot (L_q \cdot i_q - \Psi_d(\theta))$$

$$u_q = R_1 \cdot i_q + L_q \cdot \frac{di_q}{dt} + \omega_e \cdot (L_d \cdot i_d + \Psi_q(\theta))$$

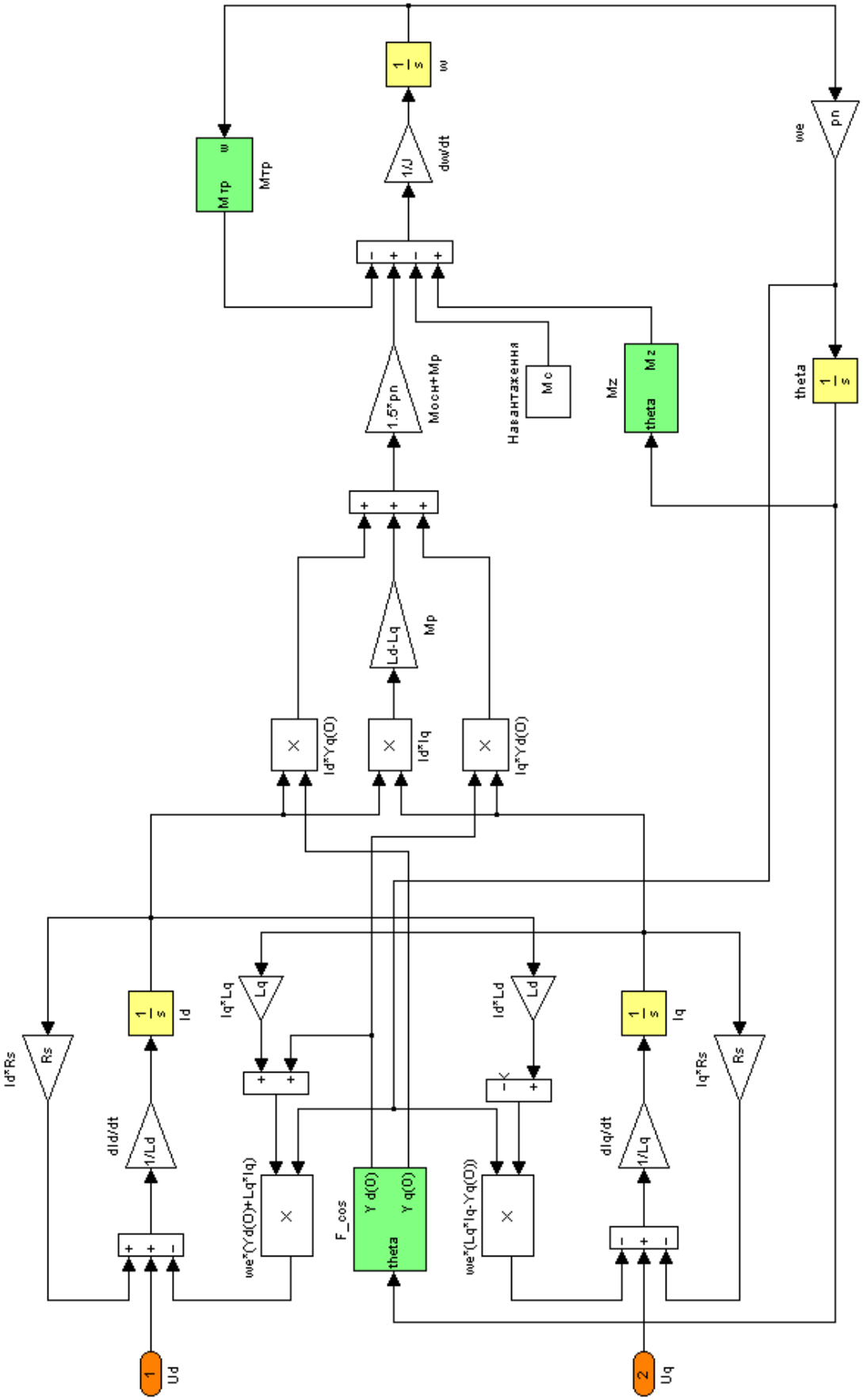
PMSM electromagnetic moment on the stator:

$$M = 1{,}5 \cdot p \cdot (\Psi_d(\theta) \cdot i_d + \Psi_q(\theta) \cdot i_q + (L_d - L_q) \cdot i_d \cdot i_q)$$

For the full description of the PMSM transients we should add main motion equation:

$$M - M_c = J \cdot \frac{d\omega}{dt}, \quad (3.23)$$

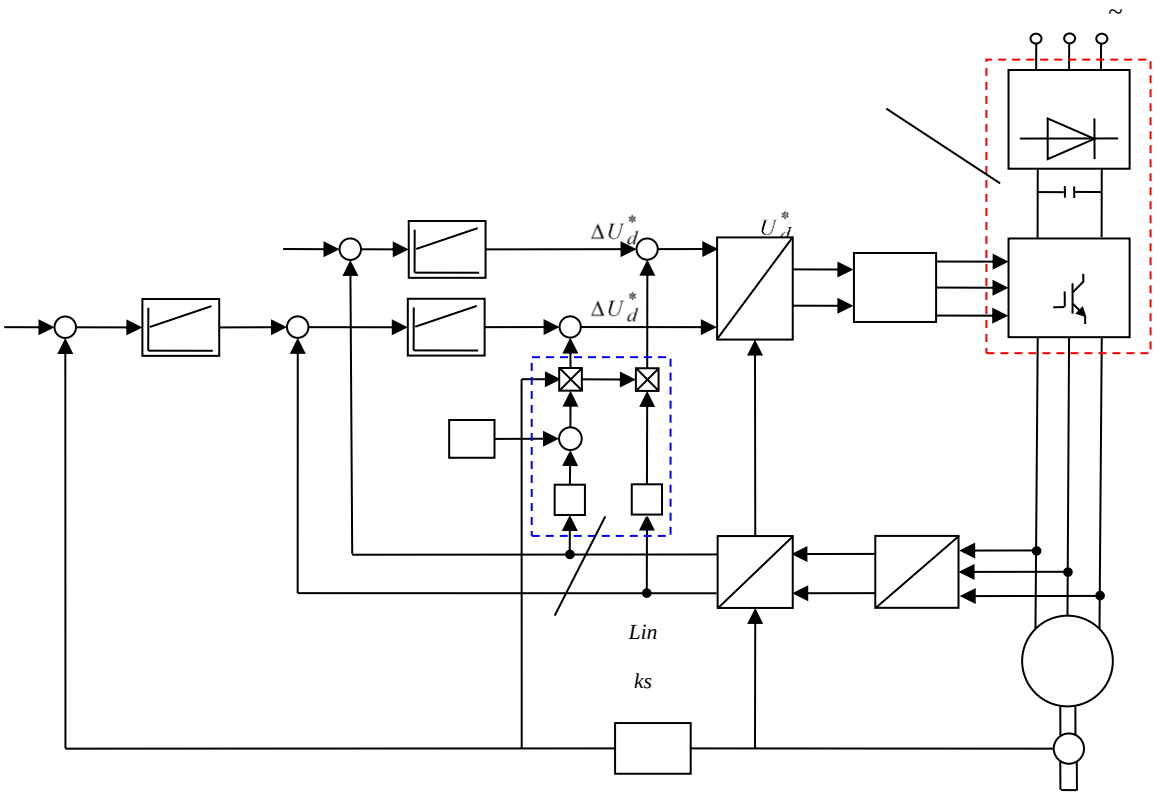
where J – PMSM moment of inertia; M_c – load torque.



					ЕП.ПД.21.201-С.StDED	
		№ докум.	Подпись			41

Simulation of PMSM drive

Electric drive is realized with classical FOC. Corresponding structure of the drive:



15 Field Oriented Control of PMSM

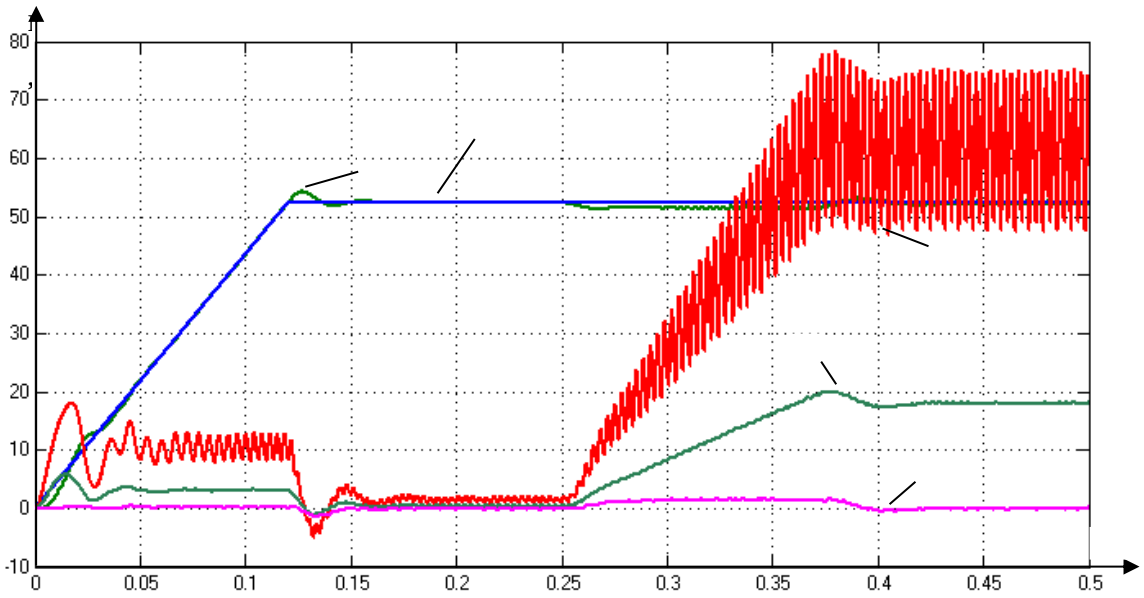
The FOC system contains of: PMSM, Rotor Position Sensor (RPS), Frequency Convertor (FC), Autonomous Voltage Inverter (AVI), Uncontrolled Rectifier (UR). There is a LC filter for filtering rectified voltage U_{dc}. Voltage is regulated by PWM with vertical control.

Task signals with this type of control method $u_d^*,\ u_q^*$ current regulators output are transformed into three sinusoidal task signals onto phase voltages of corresponding amplitude, frequency and shifted by phase by 120°. Then they equalize to the triangle base PWM voltage and in dependance of the resulting sign of the subtraction regulators send signals $s_{a,b,c}$ to the AVI.

The control system has two current loops $i_d,\ i_q$ and external speed control contour ω, where $i_q$is internal. Current regulators are set to technical optimum and speed regulator to symmetrical optimum. Speed task of electric drive is realized by the intensity setter.

Controlled values $i_d,\ i_q$ into the moving coordinates system linked, it negatively affects the dynamics of transient processes. For improving the system, independent regulation of both variables should be used. The compensation of the mutual links and EMF is applied. Compensation signals are defined after the current regulator with an opposite sign:

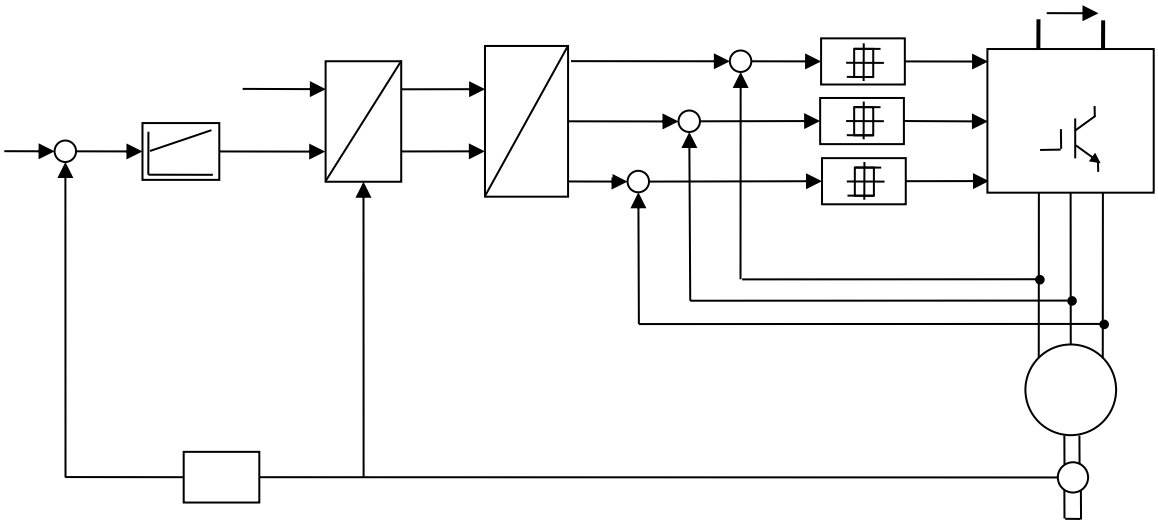
$$u_{dkom} = \omega_e \cdot (L_q \cdot i_q - \Psi_d(\theta))$$
$$u_{qkom} = \omega_e \cdot (L_d \cdot i_d + \Psi_q(\theta))$$



16 Transients during acceleration and rated load connection

Big torque and speed oscillations occur. It affects the control precision. Existing torque pulsations are caused by periodic components of the main EM torque M_{ADD}. For elimination special torque pulse compensation laws should be applied.

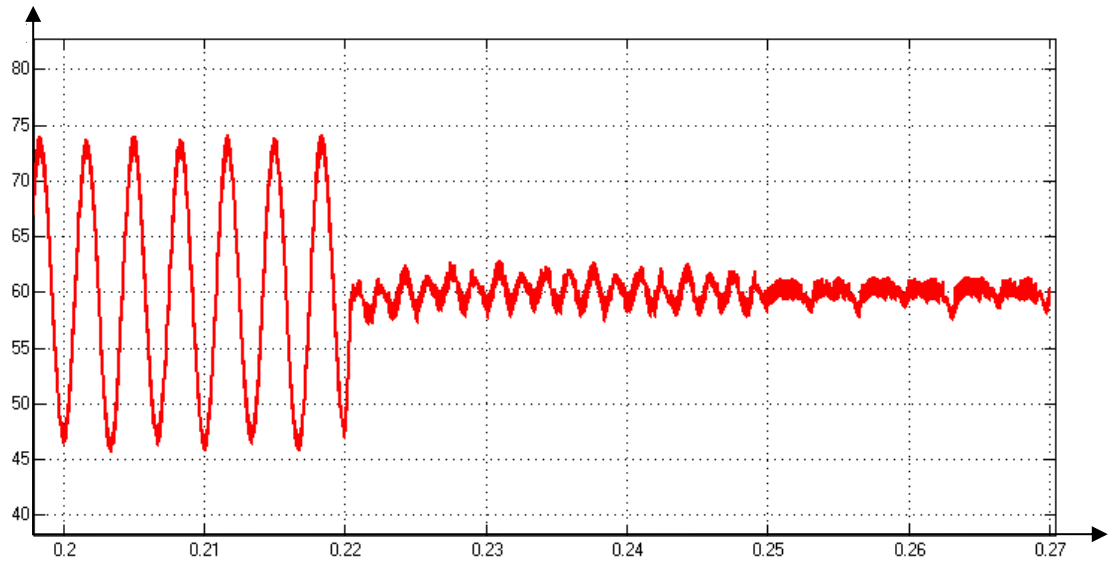
For better current harmonics current regulators must be as fast as possible. Therefore, relay current regulators usage is more meaningful. Speed regulator settings remains the same. Current range is $i^* \pm \Delta i^*$ while $\Delta i^* = 0{,}05$ V.



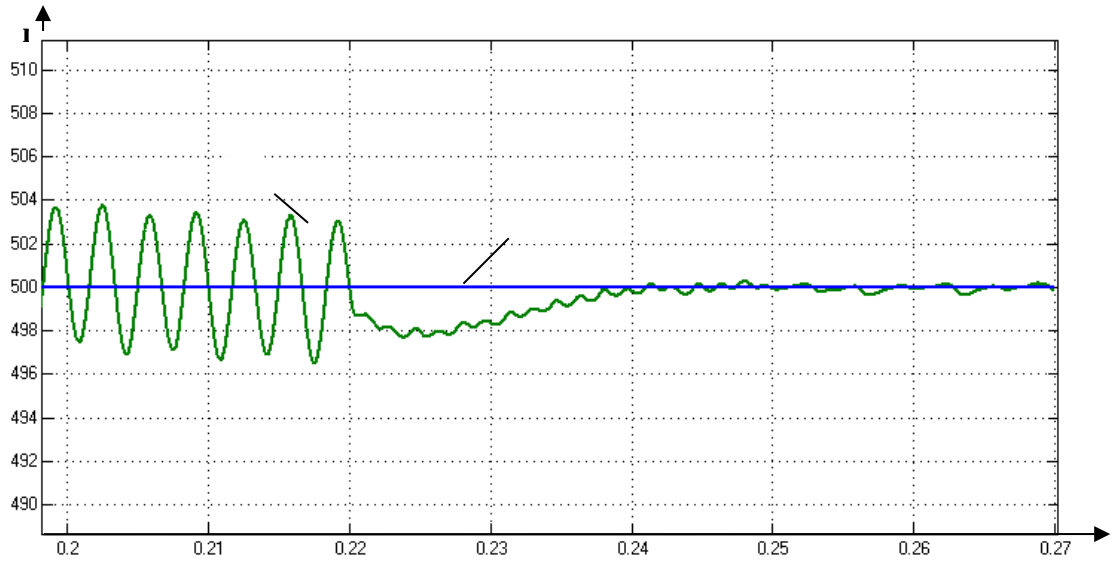
17 PMSM Drive with Relay Current Regulators

As default control method has been selected (with $i_d = 0$), reactive moment is absent. In this case the EM torque equation with consideration of flux linkage:

$$M = 1{,}5 \cdot p \cdot (i_q \cdot (\Psi_1 + \Psi_{q6} \cdot cos \ \ 6 \cdot \theta + \Psi_{q12} \cdot cos \ \ 12 \cdot \theta))$$



18 Main EM torque diagram



19 PMSM Speed Diagram

5. Техніко-Економічне Обґрунтування

Вступ

У даній роботі об’єктом розробки є датчик температури ротора (рухома частина двигуна) для СДПМ (Синхронний Двигун з Постійними Магнітами)

Через високий коефіцієнт корисної дії, компактність та широкий технологічний спектр методів керування СДПМ займає все більшу частку ринку електричних двигунів на протязі останніх 15 років.

Одним з основних елементів конструкції СДПМ є чутливі до температури постійні магніти. Для захисту від розмагнічування при високих температурах просунуті системи керування включають у собі аналітичні моделі передбачення температури в середині двигуна.

Використання аналітичних моделей обумовлено неможливістю встановлення датчиків на рухомі компоненти в середині двигуна.

Аналітичні моделі є не точними і створюють додаткове розрахункове навантаження на систему керування, що знижує її швидкодію.

Результатом науково-дослідницької роботи, що була проведена у даному проєкті є безконтактний датчик температури, що вирішує наступні проблеми:

- Датчик температури встановлюється на рухому частину двигуна (ротор)
- Датчик температури не впливає на роботу двигуна
- Датчик температури автономний і не потребує внесення значних змін у конструкцію двигуна
- Датчик є універсальним і може бути вбудованим у двигуни різних конструкцій
- Датчик температури передає набагато більш надійну інформацію щодо стану температури ротору двигуна

При впровадженні практики встановлення датчиків температури ротора у синхронні двигуни з постійними магнітами передбачувані результати наступні:

- Ріст вартості двигуна на постійну суму (частка від повної вартості двигуна не перевищуватиме 2%).
- Покращення надійності двигуна (зниження витрат на капітальний ремонт)
- Подовження строку праці (зниження амортизаційних відрахувань)

Реалізація будь-якої науково-дослідної роботи завжди вимагає витрат. Перед виробництвом нового об’єкту необхідно подбати про максимальне можливе зменшення витрат без втрати якості розробленого продукту.

Очікуваний економічний вплив впровадження результатів даної науково-дослідницької роботи є зменшення витрат на капітальний ремонт та зниження амортизаційних відрахувань, що дозволить знизити собівартість товарів вироблених на верстатах, де використовуються датчики температури ротора.

Розрахунок капітальних витрат

Для визначення капітальних витрат можна скористатися формулою

$$К_{пр} = К_{об}(\sum Ці) + З_{тзс} + З_{м} + З_{н} + З_{пр} грн.$$

де $К_{об}(\sum Ці)$ - це вартість придбання устаткування (електричний двигун, датчик, та перетворювач частоти), яке необхідно для реалізації проєкту.

$З_{тзс}$ - транспортно-заготівельні і складські витрати;

$З_{м}$ - витрати на монтажні витрати;

$З_{н}$- витрати на налагоджувальні витрати;

$З_{пр}$ - інші одноразові вкладення грошових коштів.

Табл. 5.1. Зведення капітальних витрат

№ з/п	Найменування технічних засобів (комплектуючих виробів)	Кількість	Ціна за одиницю, грн	Сума, грн
1	Двигун MOTEG MMA 80-90-80C	1	46000	46000
2	Частотний перетворювач SINAMICS V20	1	19800	19800
3	Датчик температури	1	200	200
Всього				129000

Транспортно-заготівельні і складські витрати

Транспортуванням займається стороння компанія, тому у розрахунку вказана лише сума інвойсу

$$З_{тзс} = Ц_т = 16250 \text{ грн}$$

де $Ц_т$ – ціна на транспортування двигуна від дистриб'ютора до цеху. Замовлення таких компонентів, як датчик температури виконується у локальних дистриб'юторів.

Витрати на монтажно-налагоджувальні роботи

$$З_{м(н)} = \sum (Ч_i \times а_i \times t_i) \times K_д \times K_{см} \times K_{пр}$$

$Ч_i$ – чисельність працівників і-го розряду, необхідних для виконання певного обсягу монтажних (налагоджувальних робіт), чол.;

$а_i$ – годинна тарифна ставка працівника і-го розряду, грн.;

t_i – час, необхідний для виконання певного обсягу монтажних (налагоджувальних робіт), год.;

$К_д$ – коефіцієнт, що враховує розмір доплат;

$К_{см}$ – коефіцієнт, що враховує єдиний соціальний внесок;

$К_{пр}$ – коефіцієнт, що враховує інші витрати на здійснення монтажних (налагоджувальних) робіт.

Для монтажно-налагоджувальних робіт запропоновано залучити трьох електромеханіків по засобам автоматики і приборам технологічного обладнання п’ятого розряду.

Годинна тарифна ставка працівника – 250 грн.

Час на роботи – 20 годин.

Коефіцієнт, що враховує розмір доплат – 1,25.

Коефіцієнт, що враховує єдиний соціальний внесок – 1,22.

Коефіцієнт, що враховує інші витрати – 1,05.

Отже, витрати на монтажно-налагоджувальні роботи:

$$З_{м(н)} = \sum (2 \times 250 \times 20) \times 1{,}25 \times 1{,}22 \times 1{,}05 = 16012 \text{ (грн)}$$

Проєктні капіталовкладення

$$К_{пр} = 129000 + 16250 + 16012 = 161262 \text{ грн}$$

Визначення експлуатаційних витрат

Експлуатаційні витрати - це поточні витрати на експлуатацію і обслуговування об'єкту проектування за певний період (рік), виражені в грошовій формі.

До основних статей експлуатаційних витрат по електротехнічному устаткуванню відносяться:

1. Амортизаційні відрахування (Са);

2. Заробітна плата обслуговуючого персоналу (Сз);

3. Відрахування на соціальні заходи від заробітної плати (Сс);

4. Витрати на технічне обслуговування і поточний ремонт устаткування (Ст);

5. Вартість електроенергії, споживаної об'єктом проектування (Сэ);

6. Інші експлуатаційні витрати (Сі).

Таким чином, річні експлуатаційні витрати складуть:

$$C = C_а + C_з + C_с + C_т + C_е + C_{пр}$$

Розрахунок амортизаційних відрахувань

Амортизаційні відрахування розраховані прямолінійним методом

$Т_п = 5$ років – строк корисного використання

$Л = 8000$ грн – ліквідаційна вартість основних засобів

$Ф_п$ – вартість об'єкта основних засобів

Норма амортизації 20%

$$Н_а = \frac{Ф_п - Л}{Ф_п Т_п} = \frac{161232 - 8000}{161232 \cdot 5} = 0.19 = 19.0\ \%$$

$$С_а = (Ф_п - Ф_л) \cdot Н_а = \frac{(161000 - 8000) \cdot 20}{100} = 30646\ [грн]$$

Визначення річного фонду заробітної плати

$Т_{зм} = 8$ годин – тривалість зміни

$$F_н = Д_р Т_{зм} = 205 \cdot 8 = 1640\ [годин]$$

Табл. 5.2 Розрахунок річних витрат на обслуговування та поточний ремонт

№ п/п	Найменування професій	Явочний штат у зміну, осіб	Обліковий склад з урахуванням змінності роботи, осіб	Годинна тарифна ставка	Номінальний річний фонд робочого часу, годин	Усього основна зарплата, грн
1.	Оператор верстату	2	1	225	1640	738000
Усього						738000

Додаткова заробітна плата на рік: $З_{дод} = 14350$ [грн]

Річний фонд заробітної плати:

$$С_з = З_{осн} + З_{дод} = 738000 + 14350 = 752350 \text{ [грн]}$$

Витрати на технічне обслуговування і поточний ремонт

Витрати на технічне обслуговування розраховуються за наступною формулою:

$$С_т = R \cdot t \cdot m \cdot R_\Sigma + \frac{S П}{T} T_ф$$

$R = 225$ грн – годинна ставка робітників, що виконують ремонт

$t = 1.2$ години – трудомісткість одного ремонту

$m = 2$ – число ремонтів за рік

$R_\Sigma = 2.1$ – сумарна категорія складності ремонту

$S = 240$ грн – вартість однотипних замінних елементів (мастило)

$П = 1$ – кількість однотипних замінних елементів

$Т = 1400$ годин – середній термін служби деталей даного типу

$Т_ф = 1800$ годин – число годин роботи апаратури на рік

$$С_т = 225 \cdot 1.2 \cdot 2 \cdot 2.1 + \frac{240 \cdot 1}{1400} \cdot 1800 = 1443 \text{ [грн]}$$

Розрахунок вартості спожитої електроенергії

$Ц_е = 3.8$ грн – вартість електроенергії для підприємств (2-га категорія, високовольтна мережа) за 2021 рік

$$W_p = \frac{W}{k_{втр}} \cdot Т_ф = \frac{5.5}{0.9} \cdot 1800 = 11000 \text{ [кВт]}$$

$$С_е = W_p Ц_е = 11000 \cdot 3.8 = 41800 \text{ [грн]}$$

Визначення інших витрат

$$С_{пр} = С_з \cdot 0.04 = 752350 \cdot 0.04 = 30094 \text{ [грн]}$$

Експлуатаційні витрати

$$С = 12100 + 752350 + 1443 + 41800 + 30094 = 946687 \text{ [грн]}$$

Висновки

Згідно з проведеним розрахунком ми визначили вартість капітальних витрат – 161262 грн, та вартість щорічних витрат – 946687 грн. Усі розрахунки були виконані за методичними матеріалами кафедри прикладної економіки та підприємництва НТУ «Дніпровська Політехніка» [12]

6. Охорона Праці

Аналіз безпеки під час експлуатації

Шкідливі фактори на виробництві – це небезпечні фактори робочого процесу чи умов навколишнього середовища, які можуть впливати негативно на здоров’я й працездатність людини. Тривалий вплив на людину шкідливого виробничого фактору призводить на захворювання. На промислових підприємствах, де використовуються електроприводи, існує низка шкідливих для працівника факторів:

- Вібрація під час роботи двигуна
- Шкідливий шум
- Недостатня освітленість
- Електронебезпека

Небезпечним шкідливим фактором є такий фактор, котрий призводить до травм, миттєвому погіршенню здоров’я персоналу під час впливу.

Об’єкт, який розглядається в даній роботі, має також декілька особливих небезпечних факторів, а саме:

- У складі електромеханічної системи є частини й механізми, які обертаються,
- частини електричного двигуна;
- Можливість ураження електричним струмом;
- Пожежа.

Згідно з ПУЕ (Правила улаштування електроустановок), на підприємствах, де використовують підвісні двигуни, застосовуються мережі з ізольованим нейтральним проводом. Такі засоби сприяють деяким особливостям під час аналізу небезпеки ураження електричним струмом.

Інженерно-технічні заходи з охорони праці на підприємстві

Для захисту персоналу, що обслуговує верстат, від ураження електричним струмом використовують деякі засоби захисту. До таких заходів можна віднести будь-які засоби особистого захисту: гумові рукавиці, боти й килимки, які використовують під час увімкнення(відключення) роз'єднувачів й високовольтних пристроїв, а також під час виведення електроприводу у ремонт.

Для безпеки роботи під час ремонту електричних ланцюгів необхідно відмикати пристрій від мережі живлення, удосконалитись у відсутності напруги спеціальним індикатором, виконати заземлення й встановити плакат з написом «Не вмикати! Працюють люди.» для попередження інших працівників підприємства. Усе електроустаткування повинно бути заземлене, огороджене й ізольоване. Реверсори встановлюються на висоті 2-3 метри, щоб запобігти випадкове торкання до струмопровідних частин обслуговуючим персоналом.

Усі ями, отвори у підлозі, переходи й мостики у приміщенні повинні бути огороджені перилами заввишки не менше 1 метра. Заземленню також підлягають металеві частини електроустаткувань, які зазвичай не знаходяться під напругою, але, у разі пошкодження ізоляції, можуть бути під небезпечною напругою: привода електричної апаратури, вторинні обмотки вимірювальних трансформаторів, каркаси щитів керування й розподільчих щитів, металеві та залізобетонні конструкції, металеві корпуса кабельних муфт, металеві оболонки кабелів. Для підключення заземлення на пристрої повинен бути болт. Діаметр болта не менш ніж 10 мм й напис або спеціальна наліпка біля нього «Земля». Усі з'єднання, які об'єднують «Землю», повинні бути виконані зварюванням або скріплені бовтами

Протипожежне обладнання

До складу протипожежного обладнання обов'язково входять вогнегасники (ОХВП - 10м і ОУ - 2). На нульовій позначці встановлений ящик з протипожежним інвентарем: сокири, багор, пили, ломи, лопати, відра, контейнери з піском. На відмітках 0; +8,40; +18,00; +22,80 є пожежні ідранти, які забезпечуються водою з камери пожежогасіння, розташованої на нульовій позначці. У камері пожежогасіння встановлені три насоси 4К-6 продуктивністю $135\frac{м^3}{год}$, з напором 98 метрів. Біля гідрантів знаходяться пожежні рукави з брандспойтами.

Організаційно-технічні заходи з безпеки в НС

Організаційними заходами щодо забезпечення безпеки роботи є:

- інструктаж;
- перевірка придбаних знань і навичок;
- видача нарядів і розпоряджень:
- видача дозволів на підготовку робочих місць і допуск;
- допуск;
- нагляд під час роботи;
- переведення на інше робоче місце;
- перерву в роботі та її закінчення. Технічними заходами щодо забезпечення безпеки робіт є:
- виробництво відключень;
- вивішування попереджувальних плакатів, огородження робочого місця;
- перевірка відсутності напруги, накладення заземлення.

Надзвичайна ситуація на підприємстві

Надзвичайна ситуація - це стан, при якому в результаті виникнення джерела НС на об'єкті, визначеній території або акваторії порушуються нормальні умови життя і діяльності людей, виникає загроза їх життю і здоров'ю, завдається шкода майну.

Під терміном «надзвичайна ситуація» розуміють небезпечне природне явище, аварію або небезпечну техногенну пригоду, широке поширення інфекційної хвороби людей, сільськогосподарських тварин і рослин, а також застосування сучасних засобів ураження, в результаті чого відбувається або може відбутися НС.

На підприємстві можуть статися такі аварії: вихід з ладу машин, механізмів, а також ліній електропередач. У разі якщо працівник при виконанні поставлених завдань знепритомнів. Це стан, при якому потерпілий не реагує на зовнішні подразники, при цьому дихання, пульс збережено і можна знайти, м'язи розслаблені. При такому стані людини необхідно:

- очистити дихальні шляхи;
- покласти на бік;
- повідомити начальнику бригади про подію;
- викликати швидку допомогу;
- слідувати подальшим інструкціям начальника.

Можливі випадки пошкодження кінцівок при попаданні в обертовий механізм. У даній ситуації необхідно:

- провести аварійне вимкнення чинного механізму;
- оповістити начальника про подію на місці роботи;
- слідувати подальшим вказівкам начальника;
- якщо є можливість, перенести з місця події потерпілого і надати першу медичну допомогу (обробити рану, накласти шину).

Найчастіше на підприємстві може відбуватися поразка працюючого персоналу електричним струмом. Симптоми наступні:

- у потерпілого відбувається судорожне скорочення м'язів,
- конвульсії;
- колір обличчя стає блідим;
- пульс, дихання слабшають, виникає аритмія;
- можливі втрата свідомості, зупинка дихання та серцевої діяльності.

При виникненні такого роду події необхідно:

- не торкатися до потерпілого голими руками;
- відключити джерело струму (рубильник, вимикач, розетка), якщо немає
- такої можливості, скористатися інструментом, що знаходяться в
- протипожежному ящику (сокира), і перерубати силові лінії
- електроживлення;
- повідомити старшому по об'єкту про виниклу ситуацію;
- при свідомому стані потерпілого, зігріти його і дати ліки у вигляді
- таблетки (реланиума, нозепама, фенозепама та ін.);
- очікувати прибуття медичного персоналу.

Вибір системи освітлення

Робота в приміщеннях, обладнаних промисловими механізмами неможлива без забезпечення в них оптимальних показників освітлення робочих поверхонь. Для дотримання комфортних умов праці необхідно розрахувати та обрати систему освітлення, джерело світла і світильник, визначити кількість світильників для забезпечення нормованої освітленості і розташувати їх на плані приміщення.

Тип світильників, встановлюваних у виробничих приміщеннях, обирається згідно з технологічними умовами та з урахуванням вимог до розподілу яскравості, за умовами середовища, за економічними показниками, а також з урахуванням естетичних вимог.

Для нашого випадку обрано світильники з люмінесцентними лампами.

Обираємо частково пиленепроникненний світильник ПВЛМ-Д 2х80 Вт з розмірами (1625х270х215 мм). Згідно із СНиП II-4-79, визначений розряд зорової роботи IV(а) з параметром Г = 0,8. Коефіцієнт запасу обрано 1.5 Найбільш вигідне співвідношення відстані між світильниками до розрахункової висоти підвісу дорівнює 1.4, адже крива люмінесцентного світильнику має форму косинуса (Д).

Розрахункова висота підвісу:

$$h = H - h_{\text{св}} - h_{\text{рп}} = 2.9 \text{ [м]}$$

Розраховуємо кількість світильників з люмінесцентними лампами:

відстань між рядами світильників

$$L = \lambda h = 4.75 \text{ [м]}$$

Кількість рядів світильників

$$N_p = \frac{A}{L_p} = 1$$

Кількість світильників у ряді

$$N_{\text{ср}} = \frac{10 - l_c}{l_c} = 5$$

Отже, згідно з розрахунками, загальна кількість світильників $N = N_p$,

$N_{cp} = 5$

Відстань між крайніми світильниками і стіною $l = 1.218$

Визначення коефіцієнта використання η:

$$i = \frac{AB}{h(A+B)} = 1$$

$$\rho_n = 0.7$$

$$\rho_c = 0.5$$

$$\rho_p = 0.3$$

$$\eta = 0.5$$

Розрахунок освітлення методом коефіцієнта використання

$\text{E} = 150$ лк;

Необхідний світловий потік ламп у кожному світильнику:

$$\Phi = \frac{ESkz}{N\eta} = 4950 \text{ [лм]}$$

Де E - нормована мінімальна освітленість, лк (відповідно до розряду зорової роботи), k - коефіцієнт запасу, S - освітлювана площа, м^2 ; z - коефіцієнт мінімальної освітленості, N - число світильників у приміщенні; η - коефіцієнт використання світлового потоку. Обрана лампа ЛТБ80-4 з світловим потоком 5220 лм

Усі розрахунки були виконані за методичними вказівками кафедри аерології та охорони праці НТУ «Дніпровська Політехніка» [13]

Tables of References

					ЕП.ПД.21.201-С.ToR		
Изм.	Лист	№ докум.	Підпис	Дата			
Розроб.		Біда П.П.			PMSM Drive with Rotor Temperature Sensor	Лит.	Лист 69 / Листов 124
Перевірив		Балахонцев О.В.					
Н. Контр.		Казачковський М.М.				Павло Біда	
Затв.							

Table of References

[1] General Electrics, "Report of Large Motor Reliability Survey of Industrial and Commercial Installations," IEEE, Dallas, 1985.

[2] G. Feng, C. Lai and N. C. Kar, "Speed Harmonic Based Modeling and Estimation of Permanent Magnet Temperature for PMSM Drive Using Kalman Filter," IEEE, Windsor, Canada, 2019.

[3] J. Lee and J.-I. Ha, "Temperature Estimation of PMSM Using a Difference-Estimating Feedforward Neural Network," IEEE, Seoul, South Korea, 2020.

[4] G. Feng, C. Lai and N. C. Kar, "Expectation-Maximization Particle-Filter- and Kalman-Filter-Based Permanent Magnet Temperature Estimation for PMSM Condition Monitoring Using High-Frequency Signal Injection," IEEE, Windsor, Canada, 2016.

[5] Fodorean, Popa, Minciunescu, Irimia and Szabo, "Study of a high-speed motorization for electric vehicle based on PMSM, IM and VRSM," IEEE, Cluj-Napoca, Romania, 2014.

[6] B. Thomas, "Important Characteristics of Insulated Gate Drivers," Analog Devices Inc., 2019. [Online]. Available: https://www.analog.com/media/en/technical-documentation/tech-articles/Important-Characteristics-of-Insulated-Gate-Drivers.pdf. [Accessed 7 6 2021].

[7] On Semiconductor, "RSL10: Radio SoC, Bluetooth® 5 Certified, SDK 3.5," On Semiconductor Inc., 2021. [Online]. Available: https://www.onsemi.com/products/connectivity/wireless-rf-transceivers/rsl10. [Accessed 2021].

[8] Bluetooth SIG, "Intro to Bluetooth Low Energy," Bluetooth SIG, 2021. [Online]. Available: https://www.bluetooth.com/bluetooth-resources/intro-to-bluetooth-low-energy/. [Accessed 2021].

[9] The MathWorks Inc, "MATLAB," The MathWorks Inc, 2021. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed 2021].

[10] MOTEG GmbH, 2021. [Online]. Available: https://www.moteg.de/.

[11] Siemens GmbH, "Frequency converters and variable frequency drives for every drive application," Siemens GmbH, 2021. [Online]. Available: https://new.siemens.com/global/en/products/drives/sinamics.html. [Accessed 2021].

[12] L. Tymoshenko and N. Dementieva, Методичні Вказівки до виконання економічної частини дипломної роботи, Dnipro, Ukraine: Dnipro University of Technology, 2018.

[13] V. Holinko, V. Frundin, Y. Cheberiachko and M. Ikonnikov, Методичні Рекомендації до Виконання Розділу "Охорона Праці" в Дипломних Проєктах (Роботах) Бакалаврів Інституту Електроенергетики, Dnipro, Ukraine: Dnipro University of Technology, 2012.

[14] S. Fursa, "Високоефективний Електропривод на Базі Синхронного Двигуна з Постійними Магнітами," Dnipro University of Technology, Dnipro, Ukraine, 2012.

Table of Figures

Appendix A

main.c

```
/* -----------------------------------------------------------------------------
 * This code is based on the ble_central_peripheral example code
 *
 * -----------------------------------------------------------------------------
 * app.c
 * - Main application file
 * ----------------------------------------------------------------------------- */


#include "app.h"
#include <printf.h>



int main(void)
{
    App_Initialize();


    printf_init();
    PRINTF("__peripheral_server has started!\n");


    while (1)
    {
       // Schedule all pending events.
        Kernel_Schedule();


        /* Refresh the watchdog timer */
        Sys_Watchdog_Refresh();


        /* Wait for an event before executing the scheduler again */
```

```
        SYS_WAIT_FOR_EVENT;
    }
}
```

app.h

```
/* ----------------------------------------------------------------------------
 * Copyright (c) 2015-2017 Semiconductor Components Industries, LLC (d/b/a
 * ON Semiconductor), All Rights Reserved
 *
 * Copyright (C) RivieraWaves 2009-2016
 *
 * This module is derived in part from example code provided by RivieraWaves
 * and as such the underlying code is the property of RivieraWaves [a member
 * of the CEVA, Inc. group of companies], together with additional code which
 * is the property of ON Semiconductor. The code (in whole or any part) may not
 * be redistributed in any form without prior written permission from
 * ON Semiconductor.
 *
 * The terms of use and warranty for this code are covered by contractual
 * agreements between ON Semiconductor and the licensee.
 *
 * This is Reusable Code.
 *
 * ----------------------------------------------------------------------------
 * app.h
 * - Main application header
 * ---------------------------------------------------------------------------- */

#ifndef APP_H
```

app.h

```
#define APP_H


/* ---------------------------------------------------------------------------
 * If building with a C++ compiler, make all of the definitions in this header
 * have a C binding.
 * ------------------------------------------------------------------------- */
#ifdef __cplusplus
extern "C"
{
#endif    /* ifdef __cplusplus */


/* ---------------------------------------------------------------------------
 * Include files
 * ------------------------------------------------------------------------- */
#include <rsl10.h>
#include <rsl10_ke.h>
#include <rsl10_ble.h>
#include <rsl10_profiles.h>
#include <rsl10_map_nvr.h>
#include <stdbool.h>
#include <rsl10_flash_rom.h>
#include <rsl10_protocol.h>


#include "ble_std.h"
#include "ble_custom.h"
#include "ble_bass.h"


/* ---------------------------------------------------------------------------
```

app.h

```
 * Defines
 * ---------------------------------------------------------------------------*/

/* DIO number that is used for easy re-flashing (recovery mode) */
#define RECOVERY_DIO                    12

/* DIO number that is connected to LED of EVB */
#define LED_DIO_NUM                     6

/* Output power */
#define OUTPUT_POWER_DBM                0

/* Minimum and maximum VBAT measurements */
#define VBAT_1p1V_MEASURED              0x1200
#define VBAT_1p4V_MEASURED              0x16cc

/* Maximum battery level */
#define BAT_LVL_MAX                     100

/* Set timer to 200 ms (20 times the 10 ms kernel timer resolution) */
#define TIMER_200MS_SETTING             20

/* Set timer to 2s (200 times the 10 ms kernel timer resolution) */
#define TIMER_2S_SETTING                200

typedef void (*appm_add_svc_func_t)(void);
#define DEFINE_SERVICE_ADD_FUNCTION(func) (appm_add_svc_func_t)func
#define DEFINE_MESSAGE_HANDLER(message, handler) { message, \
```

app.h

```
                                                    (ke_msg_func_t)handler }


/* List of message handlers that are used by the different profiles/services */
#define APP_MESSAGE_HANDLER_LIST                        \
    DEFINE_MESSAGE_HANDLER(APP_TEST_TIMER, APP_Timer), \
    DEFINE_MESSAGE_HANDLER(LED_TIMER, LED_Timer)


/* List of functions used to create the database */
#define SERVICE_ADD_FUNCTION_LIST                         \
    DEFINE_SERVICE_ADD_FUNCTION(Batt_ServiceAdd_Server), \
    DEFINE_SERVICE_ADD_FUNCTION(CustomService_ServiceAdd)


typedef void (*appm_enable_svc_func_t)(uint8_t);
#define DEFINE_SERVICE_ENABLE_FUNCTION(func) (appm_enable_svc_func_t)func


/* List of functions used to enable client services */
#define SERVICE_ENABLE_FUNCTION_LIST \
    DEFINE_SERVICE_ENABLE_FUNCTION(Batt_ServiceEnable_Server)


/* --------------------------------------------------------------------------
 * Global variables and types
 * --------------------------------------------------------------------------*/
extern const struct ke_task_desc TASK_DESC_APP;


/* APP Task messages */
enum appm_msg
{
    APPM_DUMMY_MSG = TASK_FIRST_MSG(TASK_ID_APP),
```

app.h

```
    /* Timer used to have a tick periodically for application */
    APP_TEST_TIMER,

    /* Timer used to control the behavior of the LED_DIO_NUM according to
     * the connection states  */
    LED_TIMER,
};

struct app_env_tag
{
    /* Battery service */
    uint8_t batt_lvl;
    uint32_t sum_batt_lvl;
    uint16_t num_batt_read;
    bool send_batt_ntf[NUM_MASTERS];
};

/* Support for the application manager and the application environment */
extern struct app_env_tag app_env;

/* List of functions used to create the database */
extern const appm_add_svc_func_t appm_add_svc_func_list[];

/* ----------------------------------------------------------------------
* Function prototype definitions
* ----------------------------------------------------------------------*/
```

app.h

```
extern int ADC_READ(void);


extern void App_Initialize(void);


extern void App_Env_Initialize(void);


extern int APP_Timer(ke_msg_id_t const msg_id, void const *param,
                     ke_task_id_t const dest_id,
                     ke_task_id_t const src_id);


extern int LED_Timer(ke_msg_id_t const msg_id, void const *param,
                     ke_task_id_t const dest_id,
                     ke_task_id_t const src_id);


extern int Msg_Handler(ke_msg_id_t const msgid, void *param,
                       ke_task_id_t const dest_id,
                       ke_task_id_t const src_id);


/* ----------------------------------------------------------------------------
 * Close the 'extern "C"' block
 * ------------------------------------------------------------------------- */
#ifdef __cplusplus
}
#endif    /* ifdef __cplusplus */


#endif    /* APP_H */
```

app_init.c

```
/* ----------------------------------------------------------------------------
 * Copyright (c) 2015-2017 Semiconductor Components Industries, LLC (d/b/a
 * ON Semiconductor), All Rights Reserved
 *
 * This code is the property of ON Semiconductor and may not be redistributed
 * in any form without prior written permission from ON Semiconductor.
 * The terms of use and warranty for this code are covered by contractual
 * agreements between ON Semiconductor and the licensee.
 *
 * This is Reusable Code.
 *
 * ----------------------------------------------------------------------------
 * app_init.c
 * - Application initialization
 * ---------------------------------------------------------------------------- */

#include "app.h"

/* Application Environment Structure */
struct app_env_tag app_env;

/* ----------------------------------------------------------------------------
 * Function      : void App_Initialize(void)
 * ----------------------------------------------------------------------------
 * Description   : Initialize the system for proper application execution
 * Inputs        : None
 * Outputs       : None
 * Assumptions   : None
```

app_init.c

```
 * ------------------------------------------------------------------------- */
void App_Initialize(void)
{
    /* Mask all interrupts */
    __set_PRIMASK(PRIMASK_DISABLE_INTERRUPTS);

    /* Disable all interrupts and clear any pending interrupts */
    Sys_NVIC_DisableAllInt();
    Sys_NVIC_ClearAllPendingInt();

    /* Test DIO12 to pause the program to make it easy to re-flash */
    DIO->CFG[RECOVERY_DIO] = DIO_MODE_INPUT  | DIO_WEAK_PULL_UP |
                             DIO_LPF_DISABLE | DIO_6X_DRIVE;
    while (DIO_DATA->ALIAS[RECOVERY_DIO] == 0);

    /* Configure the current trim settings for VCC, VDDA */
    ACS_VCC_CTRL->ICH_TRIM_BYTE = VCC_ICHTRIM_16MA_BYTE;
    ACS_VDDA_CP_CTRL->PTRIM_BYTE = VDDA_PTRIM_16MA_BYTE;

    /* Start 48 MHz XTAL oscillator */
    ACS_VDDRF_CTRL->ENABLE_ALIAS = VDDRF_ENABLE_BITBAND;
    ACS_VDDRF_CTRL->CLAMP_ALIAS = VDDRF_DISABLE_HIZ_BITBAND;

    /* Wait until VDDRF supply has powered up */
    while (ACS_VDDRF_CTRL->READY_ALIAS != VDDRF_READY_BITBAND);

    /* Disable RF TX power amplifier supply voltage and
    * connect the switched output to VDDRF regulator */
```

app_init.c

```
    ACS_VDDPA_CTRL->ENABLE_ALIAS = VDDPA_DISABLE_BITBAND;
    ACS_VDDPA_CTRL->VDDPA_SW_CTRL_ALIAS = VDDPA_SW_VDDRF_BITBAND;


    /* Enable RF power switches */
    SYSCTRL_RF_POWER_CFG->RF_POWER_ALIAS = RF_POWER_ENABLE_BITBAND;


    /* Remove RF isolation */
    SYSCTRL_RF_ACCESS_CFG->RF_ACCESS_ALIAS = RF_ACCESS_ENABLE_BITBAND;


    /* Start the 48 MHz oscillator without changing the other register bits */
    RF->XTAL_CTRL = ((RF->XTAL_CTRL & ~XTAL_CTRL_DISABLE_OSCILLATOR) |
                     XTAL_CTRL_REG_VALUE_SEL_INTERNAL);


    /* Enable the 48 MHz oscillator divider using the desired prescale value */
    RF_REG2F->CK_DIV_1_6_CK_DIV_1_6_BYTE = CK_DIV_1_6_PRESCALE_6_BYTE;


    /* Wait until 48 MHz oscillator is started */
    while (RF_REG39->ANALOG_INFO_CLK_DIG_READY_ALIAS !=
           ANALOG_INFO_CLK_DIG_READY_BITBAND);


    /* Switch to (divided 48 MHz) oscillator clock */
    Sys_Clocks_SystemClkConfig(JTCK_PRESCALE_1   |
                               EXTCLK_PRESCALE_1 |
                               SYSCLK_CLKSRC_RFCLK);


    /* Configure clock dividers */
    CLK->DIV_CFG0 = (SLOWCLK_PRESCALE_8 | BBCLK_PRESCALE_1 |
                     USRCLK_PRESCALE_1);
```

app_init.c

```
    CLK->DIV_CFG2 = (CPCLK_PRESCALE_8 | DCCLK_PRESCALE_2);


    BBIF->CTRL = (BB_CLK_ENABLE | BBCLK_DIVIDER_8 | BB_WAKEUP);


    /* Configure ADC channel 0 to measure VBAT/2 */
    /*Sys_ADC_Set_Config(ADC_VBAT_DIV2_NORMAL | ADC_NORMAL |
                        ADC_PRESCALE_6400);*/
    Sys_ADC_InputSelectConfig(5,
                              (ADC_NEG_INPUT_GND |
                               ADC_POS_INPUT_VBAT_DIV2));


    Sys_DIO_Config(0, DIO_MODE_GPIO_OUT_1);


    Sys_DIO_Config(1, DIO_MODE_GPIO_IN_0);
    Sys_ADC_InputSelectConfig(1,
                              (ADC_NEG_INPUT_GND |
                                        ADC_POS_INPUT_DIO1));
    Sys_DIO_Config(3, DIO_MODE_GPIO_IN_0);
    Sys_ADC_InputSelectConfig(3,
                              (ADC_NEG_INPUT_GND |
                                        ADC_POS_INPUT_DIO3));
    /* Configure DIOs */
    Sys_DIO_Config(LED_DIO_NUM, DIO_MODE_GPIO_OUT_0);


    /* Initialize the baseband and BLE stack */
    BLE_Initialize();


    /* Set radio output power of RF */
```

app_init.c

```
    Sys_RFFE_SetTXPower(OUTPUT_POWER_DBM);


    /* Initialize environment */
    App_Env_Initialize();


    /* Stop masking interrupts */
    __set_PRIMASK(PRIMASK_ENABLE_INTERRUPTS);
    __set_FAULTMASK(FAULTMASK_ENABLE_INTERRUPTS);
}


/* ----------------------------------------------------------------------------
 * Function      : void App_Env_Initialize(void)
 * ----------------------------------------------------------------------------
 * Description   : Initialize application environment
 * Inputs        : None
 * Outputs       : None
 * Assumptions   : None
 * ---------------------------------------------------------------------------- */
void App_Env_Initialize(void)
{
    /* Reset the application manager environment */
    memset(&app_env, 0, sizeof(app_env));


    /* Create the application task handler */
    ke_task_create(TASK_APP, &TASK_DESC_APP);


    /* Initialize the custom service environment */
    CustomService_Env_Initialize();
```

app_init.c

```
    /* Initialize the battery service server environment */
    Bass_Env_Initialize();
}
```

ble_custom.c

```
/* ----------------------------------------------------------------------------
 * Copyright (c) 2015-2017 Semiconductor Components Industries, LLC (d/b/a
 * ON Semiconductor), All Rights Reserved
 *
 * This code is the property of ON Semiconductor and may not be redistributed
 * in any form without prior written permission from ON Semiconductor.
 * The terms of use and warranty for this code are covered by contractual
 * agreements between ON Semiconductor and the licensee.
 *
 * This is Reusable Code.
 *
 * ----------------------------------------------------------------------------
 * ble_custom.c
 * - Bluetooth custom service functions
 * ---------------------------------------------------------------------------- */

#include "app.h"
#include <printf.h>

/* Global variable definition cs_env[i]. */
struct cs_env_tag cs_env[NUM_MASTERS];
const uint32_t cs_atts_len[] = {
```

ble_custom.c

```
    [CS_IDX_TX_VALUE_VAL] = CS_TX_VALUE_MAX_LENGTH,
    [CS_IDX_RX_VALUE_VAL] = CS_RX_VALUE_MAX_LENGTH,
    [CS_IDX_TX_LONG_VALUE_VAL] = CS_TX_LONG_VALUE_MAX_LENGTH,
    [CS_IDX_RX_LONG_VALUE_VAL] = CS_RX_LONG_VALUE_MAX_LENGTH,
};



int ADC_READ(void)
{

      /*Sys_ADC_Set_Config(ADC_VBAT_DIV2_NORMAL | ADC_NORMAL |
                           ADC_PRESCALE_6400);*/

      ADC->CFG = (ADC_PRESCALE_200 & ((1U << ADC_CFG_DUTY_DIVIDER_Pos)   |
                           /*(1U << ADC_CFG_CONTINUOUS_MODE_Pos )          |*/
                           ADC_CFG_FREQ_Mask));

      uint32_t v_batt = (ADC->DATA_TRIM_CH[5] * 2);      //    V_batt
      uint32_t v_r31 = ADC->DATA_TRIM_CH[1];       // V_R1 module 1
      uint32_t v_r33 = ADC->DATA_TRIM_CH[3];       // V_R1 module 3

      for (unsigned int i = 0; i < NUM_MASTERS; i++)
      {
            cs_env[i].tx_value[0] = (v_batt >> 24) & 0xFF;
            cs_env[i].tx_value[1] = (v_batt >> 16) & 0xFF;
            cs_env[i].tx_value[2] = (v_batt >> 8) & 0xFF;
            cs_env[i].tx_value[3] = v_batt & 0xFF;
```

ble_custom.c

```
            cs_env[i].tx_value[4] = (v_r31 >> 24) & 0xFF;
            cs_env[i].tx_value[5] = (v_r31 >> 16) & 0xFF;
            cs_env[i].tx_value[6] = (v_r31 >> 8) & 0xFF;
            cs_env[i].tx_value[7] = v_r31 & 0xFF;

            cs_env[i].tx_value[8] = (v_r33 >> 24) & 0xFF;
            cs_env[i].tx_value[9] = (v_r33 >> 16) & 0xFF;
            cs_env[i].tx_value[10]= (v_r33 >> 8) & 0xFF;
            cs_env[i].tx_value[11]= v_r33 & 0xFF;

            for(uint8_t o = 12; o < CS_TX_VALUE_MAX_LENGTH; o++)
                  cs_env[i].tx_value[o] = 0;
      }

      return 1;
}

/* ----------------------------------------------------------------------------
 * Function      : void CustomService_Env_Initialize(void)
 * ----------------------------------------------------------------------------
 * Description   : Initialize custom service environment
 * Inputs        : None
 * Outputs       : None
 * Assumptions   : None
 * ---------------------------------------------------------------------------- */
void CustomService_Env_Initialize(void)
{
```

ble_custom.c

```
       PRINTF("!!! CustomService_Env_Initialize\n");
    for (unsigned int i = 0; i < NUM_MASTERS; i++)
    {
        /* Reset the application manager environment */
        memset(&cs_env[i], 0, sizeof(struct cs_env_tag));
        cs_env[i].sent_success = true;
        cs_env[i].tx_cccd_value = ATT_CCC_START_NTF;
        cs_env[i].rx_cccd_value = 0;
    }
}

/* ----------------------------------------------------------------------------
 * Function      : void CustomService_ServiceAdd(void)
 * ----------------------------------------------------------------------------
 * Description   : Send request to add custom profile into the attribute
 *                 database.Defines the different access functions
 *                 (setter/getter commands to access the different
 *                 characteristic attributes).
 * Inputs        : None
 * Outputs       : None
 * Assumptions   : None
 * ---------------------------------------------------------------------------- */
void CustomService_ServiceAdd(void)
{
    struct gattm_add_svc_req *req =
        KE_MSG_ALLOC_DYN(GATTM_ADD_SVC_REQ,
                         TASK_GATTM, TASK_APP,
                         gattm_add_svc_req,
```

ble_custom.c

```
                        CS_IDX_NB * sizeof(struct gattm_att_desc));

    const uint8_t svc_uuid[ATT_UUID_128_LEN] = CS_SVC_UUID;

    const struct gattm_att_desc att[CS_IDX_NB] =
    {
        /* Attribute Index  = Attribute properties: UUID,
         *                                          Permissions,
         *                                          Max size,
         *                                          Extra permissions */

        /* TX Characteristic */
        [CS_IDX_TX_VALUE_CHAR]     = ATT_DECL_CHAR(),

        [CS_IDX_TX_VALUE_VAL]      =
ATT_DECL_CHAR_UUID_128(CS_CHARACTERISTIC_TX_UUID,
                                                              PERM(RD, ENABLE) |
PERM(NTF, ENABLE),

CS_TX_VALUE_MAX_LENGTH),

        [CS_IDX_TX_VALUE_CCC]      = ATT_DECL_CHAR_CCC(),

        [CS_IDX_TX_VALUE_USR_DSCP] =
ATT_DECL_CHAR_USER_DESC(CS_USER_DESCRIPTION_MAX_LENGTH),

        /* RX Characteristic */
        [CS_IDX_RX_VALUE_CHAR]     = ATT_DECL_CHAR(),
```

ble_custom.c

```
        [CS_IDX_RX_VALUE_VAL]      =
ATT_DECL_CHAR_UUID_128(CS_CHARACTERISTIC_RX_UUID,
                                                                PERM(RD, ENABLE) |
PERM(WRITE_REQ, ENABLE)
                                                                | PERM(WRITE_COMMAND,
ENABLE),

CS_RX_VALUE_MAX_LENGTH),

        [CS_IDX_RX_VALUE_CCC]      = ATT_DECL_CHAR_CCC(),
        [CS_IDX_RX_VALUE_USR_DSCP] =
ATT_DECL_CHAR_USER_DESC(CS_USER_DESCRIPTION_MAX_LENGTH),

        /* TX long Characteristic */
        [CS_IDX_TX_LONG_VALUE_CHAR] = ATT_DECL_CHAR(),

        [CS_IDX_TX_LONG_VALUE_VAL]  =
ATT_DECL_CHAR_UUID_128(CS_CHARACTERISTIC_TX_LONG_UUID,
                                                                PERM(RD, ENABLE),
CS_TX_LONG_VALUE_MAX_LENGTH),

        [CS_IDX_TX_LONG_VALUE_USR_DSCP] =
ATT_DECL_CHAR_USER_DESC(CS_USER_DESCRIPTION_MAX_LENGTH),

        /* RX long Characteristic */
        [CS_IDX_RX_LONG_VALUE_CHAR]     = ATT_DECL_CHAR(),
```

ble_custom.c

```
        [CS_IDX_RX_LONG_VALUE_VAL]      =
ATT_DECL_CHAR_UUID_128(CS_CHARACTERISTIC_RX_LONG_UUID,
                                                                PERM(RD, ENABLE) |
PERM(WRITE_REQ, ENABLE)
                                                                |
PERM(WRITE_COMMAND, ENABLE),

CS_RX_LONG_VALUE_MAX_LENGTH),

        [CS_IDX_RX_LONG_VALUE_USR_DSCP] =
ATT_DECL_CHAR_USER_DESC(CS_USER_DESCRIPTION_MAX_LENGTH),
    };

    /* Fill the add custom service message */
    req->svc_desc.start_hdl = 0;
    req->svc_desc.task_id = TASK_APP;
    req->svc_desc.perm = PERM(SVC_UUID_LEN, UUID_128);
    req->svc_desc.nb_att = CS_IDX_NB;

    memcpy(&req->svc_desc.uuid[0], &svc_uuid[0], ATT_UUID_128_LEN);

    for (unsigned int i = 0; i < CS_IDX_NB; i++)
    {
        memcpy(&req->svc_desc.atts[i], &att[i],
               sizeof(struct gattm_att_desc));
    }

    /* Send the message */
```

ble_custom.c

```
    ke_msg_send(req);
}


/* ----------------------------------------------------------------------
 * Function      : int GATTM_AddSvcRsp(ke_msg_id_t const msg_id,
 *                                     struct gattm_add_svc_rsp
 *                                     const *param,
 *                                     ke_task_id_t const dest_id,
 *                                     ke_task_id_t const src_id)
 * ----------------------------------------------------------------------
 * Description   : Handle the response from adding a service in the attribute
 *                 database from the GATT manager
 * Inputs        : - msg_id     - Kernel message ID number
 *                 - param      - Message parameters in format of
 *                                struct gattm_add_svc_rsp
 *                 - dest_id    - Destination task ID number
 *                 - src_id     - Source task ID number
 * Outputs       : return value - Indicate if the message was consumed;
 *                                compare with KE_MSG_CONSUMED
 * Assumptions   : None
 * ---------------------------------------------------------------------- */
int GATTM_AddSvcRsp(ke_msg_id_t const msg_id,
                    struct gattm_add_svc_rsp const *param,
                    ke_task_id_t const dest_id,
                    ke_task_id_t const src_id)
{
    for (unsigned int i = 0; i < NUM_MASTERS; i++)
    {
```

ble_custom.c

```
        cs_env[i].start_hdl = param->start_hdl;
    }

    PRINTF("__CUSTOM SERVICE ADDED\n");
    /* Add the next requested service  */
    if (!Service_Add())
    {
        /* All services have been added, go to the ready state
         * and start advertising */
        for (unsigned int i = 0; i < NUM_MASTERS; i++)
        {
            ble_env[i].state = APPM_READY;
        }
        Advertising_Start();
    }

    return (KE_MSG_CONSUMED);
}

/* ----------------------------------------------------------------------------
 * Function      : int GATTC_ReadReqInd(ke_msg_id_t const msg_id,
 *                                      struct gattc_read_req_ind
 *                                      const *param,
 *                                      ke_task_id_t const dest_id,
 *                                      ke_task_id_t const src_id)
 * ----------------------------------------------------------------------------
 * Description   : Handle received read request indication
 *                 from a GATT Controller
```

ble_custom.c

```
 * Inputs        : - msg_id     - Kernel message ID number
 *                 - param      - Message parameters in format of
 *                                struct gattc_read_req_ind
 *                 - dest_id    - Destination task ID number
 *                 - src_id     - Source task ID number
 * Outputs       : return value - Indicate if the message was consumed;
 *                                compare with KE_MSG_CONSUMED
 * Assumptions   : None
 * ---------------------------------------------------------------------- */
int GATTC_ReadReqInd(ke_msg_id_t const msg_id,
                     struct gattc_read_req_ind const *param,
                     ke_task_id_t const dest_id,
                     ke_task_id_t const src_id)
{
    uint8_t length = 0;
    uint8_t status = GAP_ERR_NO_ERROR;
    uint16_t attnum;
    uint8_t *valptr = NULL;

    /* Retrieve the index of environment structure representing peer device */
    signed int device_indx = Find_Connected_Device_Index(KE_IDX_GET(src_id));

    if (device_indx == INVALID_DEV_IDX)
    {
        return (KE_MSG_CONSUMED);
    }

    struct gattc_read_cfm *cfm;
```

ble_custom.c

```
    /* Set the attribute handle using the attribute index
     * in the custom service */
    if (param->handle > cs_env[device_indx].start_hdl)
    {
        attnum = (param->handle - cs_env[device_indx].start_hdl - 1);
    }
    else
    {
        status = ATT_ERR_INVALID_HANDLE;
    }

    /* If there is no error, send back the requested attribute value */
    if (status == GAP_ERR_NO_ERROR)
    {
        switch (attnum)
        {
            case CS_IDX_RX_VALUE_VAL:
            {
                length = CS_RX_VALUE_MAX_LENGTH;
                valptr = (uint8_t *)&cs_env[device_indx].rx_value;
            }
            break;

            case CS_IDX_RX_VALUE_CCC:
            {
                length = 2;
                valptr = (uint8_t *)&cs_env[device_indx].rx_cccd_value;
```

ble_custom.c

```
            }
            break;

            case CS_IDX_RX_VALUE_USR_DSCP:
            {
                length = strlen(CS_RX_CHARACTERISTIC_NAME);
                valptr = (uint8_t *)CS_RX_CHARACTERISTIC_NAME;
            }
            break;

            case CS_IDX_TX_VALUE_VAL:
            {
             ADC_READ();
                length = CS_TX_VALUE_MAX_LENGTH;
                valptr = (uint8_t *)&cs_env[device_indx].tx_value;
            }
            break;

            case CS_IDX_TX_VALUE_CCC:
            {
                length = 2;
                valptr = (uint8_t *)&cs_env[device_indx].tx_cccd_value;
            }
            break;

            case CS_IDX_TX_VALUE_USR_DSCP:
            {
                length = strlen(CS_TX_CHARACTERISTIC_NAME);
```

ble_custom.c

```
                valptr = (uint8_t *)CS_TX_CHARACTERISTIC_NAME;
            }
            break;

            /* RX long characteristic*/
            case CS_IDX_RX_LONG_VALUE_VAL:
            {
                length = CS_RX_LONG_VALUE_MAX_LENGTH;
                valptr = (uint8_t *)&cs_env[device_indx].rx_long_value;
            }
            break;

            case CS_IDX_RX_LONG_VALUE_USR_DSCP:
            {
                length = strlen(CS_RX_LONG_CHARACTERISTIC_NAME);
                valptr = (uint8_t *)CS_RX_LONG_CHARACTERISTIC_NAME;
            }
            break;

            /* TX long characteristic */
            case CS_IDX_TX_LONG_VALUE_VAL:
            {
                length = CS_TX_LONG_VALUE_MAX_LENGTH;
                valptr = (uint8_t *)&cs_env[device_indx].tx_long_value;
            }
            break;

            case CS_IDX_TX_LONG_VALUE_USR_DSCP:
```

ble_custom.c

```
            {
                length = strlen(CS_TX_LONG_CHARACTERISTIC_NAME);
                valptr = (uint8_t *)CS_TX_LONG_CHARACTERISTIC_NAME;
            }
            break;

            default:
            {
                status = ATT_ERR_READ_NOT_PERMITTED;
            }
            break;
        }
    }

    /* Allocate and build message */
    cfm = KE_MSG_ALLOC_DYN(GATTC_READ_CFM,
                           KE_BUILD_ID(TASK_GATTC, ble_env[device_indx].conidx),
                           TASK_APP,
                           gattc_read_cfm, length);

    if (valptr != NULL)
    {
        memcpy(cfm->value, valptr, length);
    }

    cfm->handle = param->handle;
    cfm->length = length;
    cfm->status = status;
```

ble_custom.c

```
    /* Send the message */
    ke_msg_send(cfm);

    return (KE_MSG_CONSUMED);
}

/* ----------------------------------------------------------------------
 * Function      : int GATTC_WriteReqInd(ke_msg_id_t const msg_id,
 *                                       struct gattc_write_req_ind
 *                                       const *param,
 *                                       ke_task_id_t const dest_id,
 *                                       ke_task_id_t const src_id)
 * ----------------------------------------------------------------------
 * Description   : Handle received write request indication
 *                 from a GATT Controller
 * Inputs        : - msg_id     - Kernel message ID number
 *                 - param      - Message parameters in format of
 *                                struct gattc_write_req_ind
 *                 - dest_id    - Destination task ID number
 *                 - src_id     - Source task ID number
 * Outputs       : return value - Indicate if the message was consumed;
 *                                compare with KE_MSG_CONSUMED
 * Assumptions   : None
 * ---------------------------------------------------------------------- */
int GATTC_WriteReqInd(ke_msg_id_t const msg_id,
                      struct gattc_write_req_ind const *param,
                      ke_task_id_t const dest_id,
```

ble_custom.c

```
                        ke_task_id_t const src_id)
{
    /* Retrieve the index of environment structure representing peer device */
    signed int device_indx = Find_Connected_Device_Index(KE_IDX_GET(src_id));

    if (device_indx == INVALID_DEV_IDX)
    {
        return (KE_MSG_CONSUMED);
    }

    struct gattc_write_cfm *cfm =
        KE_MSG_ALLOC(GATTC_WRITE_CFM,
                     KE_BUILD_ID(TASK_GATTC, ble_env[device_indx].conidx),
                     TASK_APP, gattc_write_cfm);

    uint8_t status = GAP_ERR_NO_ERROR;
    uint16_t attnum;
    uint8_t *valptr = NULL;

    /* Check that offset is not zero */
    if (param->offset)
    {
        status = ATT_ERR_INVALID_OFFSET;
    }

    /* Set the attribute handle using the attribute index
     * in the custom service */
    if (param->handle > cs_env[device_indx].start_hdl)
```

ble_custom.c

```
    {
        attnum = (param->handle - cs_env[device_indx].start_hdl - 1);
    }
    else
    {
        status = ATT_ERR_INVALID_HANDLE;
    }

    /* If there is no error, save the requested attribute value */
    if (status == GAP_ERR_NO_ERROR)
    {
        switch (attnum)
        {
            case CS_IDX_RX_VALUE_VAL:
            {
                valptr = (uint8_t *)&cs_env[device_indx].rx_value;
                cs_env[device_indx].rx_value_changed = 1;
            }
            break;

            case CS_IDX_RX_VALUE_CCC:
            {
                valptr = (uint8_t *)&cs_env[device_indx].rx_cccd_value;
            }
            break;

            case CS_IDX_TX_VALUE_CCC:
            {
```

ble_custom.c

```
                valptr = (uint8_t *)&cs_env[device_indx].tx_cccd_value;
            }
            break;

            case CS_IDX_RX_LONG_VALUE_VAL:
            {
                valptr = (uint8_t *)&cs_env[device_indx].rx_long_value;
                cs_env[device_indx].rx_long_value_changed = 1;
            }
            break;

            default:
            {
                status = ATT_ERR_WRITE_NOT_PERMITTED;
            }
            break;
        }
    }

    if (valptr != NULL)
    {
        memcpy(valptr, param->value, param->length);
    }

    cfm->handle = param->handle;
    cfm->status = status;

    /* Send the message */
```

ble_custom.c

```
    ke_msg_send(cfm);


    return (KE_MSG_CONSUMED);
}


/* ----------------------------------------------------------------------
 * Function      : void CustomService_SendNotification(uint8_t conidx,
 *                                uint8_t attidx, uint8_t *value, uint8_t length)
 * ----------------------------------------------------------------------
 * Description   : Send a notification to the client device
 * Inputs        : - conidx       - connection index
 *                 - attidx       - index to attributes in the service
 *                 - value        - pointer to value
 *                 - length       - length of value
 * Outputs       : None
 * Assumptions   : None
 * ---------------------------------------------------------------------- */
void CustomService_SendNotification(uint8_t conidx, uint8_t attidx,
                                    uint8_t *value, uint8_t length)
{
    struct gattc_send_evt_cmd *cmd;


    /* Retrieve the index of environment structure representing peer device */
    signed int device_indx = Find_Connected_Device_Index(conidx);


    uint16_t handle = (attidx + cs_env[device_indx].start_hdl + 1);


    if (device_indx == INVALID_DEV_IDX)
```

ble_custom.c

```
    {
        return;
    }

    /* Prepare a notification message for the specified attribute */
    cmd = KE_MSG_ALLOC_DYN(GATTC_SEND_EVT_CMD,
                           KE_BUILD_ID(TASK_GATTC, conidx), TASK_APP,
                           gattc_send_evt_cmd,
                           length * sizeof(uint8_t));
    cmd->handle = handle;
    cmd->length = length;
    cmd->operation = GATTC_NOTIFY;
    cmd->seq_num = 0;
    memcpy(cmd->value, value, length);

    cs_env[device_indx].sent_success = false;

    /* Send the message */
    ke_msg_send(cmd);
}

/* ----------------------------------------------------------------------------
 * Function      : int GATTC_CmpEvt(ke_msg_id_t const msg_id,
 *                                  struct gattc_cmp_evt
 *                                  const *param,
 *                                  ke_task_id_t const dest_id,
 *                                  ke_task_id_t const src_id)
 * ----------------------------------------------------------------------------
```

ble_custom.c

```
 * Description   : Handle received GATT controller complete event
 * Inputs        : - msg_id     - Kernel message ID number
 *                 - param      - Message parameters in format of
 *                                struct gattc_cmp_evt
 *                 - dest_id    - Destination task ID number
 *                 - src_id     - Source task ID number
 * Outputs       : return value - Indicate if the message was consumed;
 *                                compare with KE_MSG_CONSUMED
 * Assumptions   : None
 * ---------------------------------------------------------------------- */
int GATTC_CmpEvt(ke_msg_id_t const msg_id,
                 struct gattc_cmp_evt const *param,
                 ke_task_id_t const dest_id, ke_task_id_t const src_id)
{
    /* Retrieve the index of environment structure representing peer device */
    signed int device_indx = Find_Connected_Device_Index(KE_IDX_GET(src_id));


    if (device_indx == INVALID_DEV_IDX)
    {
        return (KE_MSG_CONSUMED);
    }


    if (param->operation == GATTC_NOTIFY)
    {
        if (param->status == GAP_ERR_NO_ERROR ||
            param->status == GAP_ERR_DISCONNECTED)
        {
            cs_env[device_indx].sent_success = true;
```

```
        }
    }

    return (KE_MSG_CONSUMED);
}

/* ----------------------------------------------------------------------
 * Function      : int GATTC_AttInfoReqInd(
 *                                 ke_msg_id_t const msg_id,
 *                                 struct gattc_read_req_ind const *param,
 *                                 ke_task_id_t const dest_id,
 *                                 ke_task_id_t const src_id)
 * ----------------------------------------------------------------------
 * Description   : Request Attribute info to upper layer
 *                 could be trigger during prepare write to check if attribute
 *                 modification is authorized by profile/application or not and
 *                 to get current attribute length
 * Inputs        : - msg_id     - Kernel message ID number
 *                 - param      - Message parameters in format of
 *                                struct gattc_read_req_ind
 *                 - dest_id    - Destination task ID number
 *                 - src_id     - Source task ID number
 * Outputs       : return value - Indicate if the message was consumed;
 *                                compare with KE_MSG_CONSUMED
 * Assumptions   : None
 * ---------------------------------------------------------------------- */

int GATTC_AttInfoReqInd(ke_msg_id_t const msg_id,
```

ble_custom.c

```
                        struct gattc_read_req_ind const *param,
                        ke_task_id_t const dest_id,
                        ke_task_id_t const src_id)
{
    signed int device_indx = Find_Connected_Device_Index(KE_IDX_GET(src_id));
    if (device_indx == INVALID_DEV_IDX)
    {
        return (KE_MSG_CONSUMED);
    }

    uint16_t attnum;
    attnum = (param->handle - cs_env[device_indx].start_hdl - 1);

    /* Attribute Information confirmation message to inform if peer
     * device is authorized to modify attribute value, and to get current
     * attribute length.
     */
    struct gattc_att_info_cfm *cfm =
        KE_MSG_ALLOC(GATTC_ATT_INFO_CFM,
                     KE_BUILD_ID(TASK_GATTC, ble_env[device_indx].conidx),
                     TASK_APP, gattc_att_info_cfm);

    if (attnum == CS_IDX_RX_VALUE_VAL ||
        attnum == CS_IDX_RX_LONG_VALUE_VAL)
    {
        cfm->handle = param->handle;
        cfm->length = cs_atts_len[attnum];
        cfm->status = GAP_ERR_NO_ERROR;
```

ble_custom.c

```
        /* Send the message */
        ke_msg_send(cfm);
    }

    return (KE_MSG_CONSUMED);
}
```

ble_custom.h

```
/* ----------------------------------------------------------------------------
 * Copyright (c) 2015-2017 Semiconductor Components Industries, LLC (d/b/a
 * ON Semiconductor), All Rights Reserved
 *
 * This code is the property of ON Semiconductor and may not be redistributed
 * in any form without prior written permission from ON Semiconductor.
 * The terms of use and warranty for this code are covered by contractual
 * agreements between ON Semiconductor and the licensee.
 *
 * This is Reusable Code.
 *
 * ----------------------------------------------------------------------------
 * ble_custom.h
 * - Bluetooth custom service header
 * ---------------------------------------------------------------------------- */

#ifndef BLE_CUSTOM_H
#define BLE_CUSTOM_H
```

ble_custom.h

```
/* ---------------------------------------------------------------------------
 * If building with a C++ compiler, make all of the definitions in this header
 * have a C binding. 0xe0
 * ------------------------------------------------------------------------- */
#ifdef __cplusplus
extern "C"
{
#endif    /* ifdef __cplusplus */


/* ---------------------------------------------------------------------------
 * Include files
 * ------------------------------------------------------------------------- */


/* ---------------------------------------------------------------------------
 * Defines
 * ------------------------------------------------------------------------- */


/* Custom service UUIDs */
#define CS_SVC_UUID                     { 0x24, 0xdc, 0x0e, 0x6e, 0x01, 0x40, \
                                          0xca, 0x9e, 0xe5, 0xa9, 0xa3, 0x00, \
                                          0xb5, 0xf3, 0x93, 0xe0 }
#define CS_CHARACTERISTIC_TX_UUID       { 0x24, 0xdc, 0x0e, 0x6e, 0x02, 0x40, \
                                          0xca, 0x9e, 0xe5, 0xa9, 0xa3, 0x00, \
                                          0xb5, 0xf3, 0x93, 0xe0 }
#define CS_CHARACTERISTIC_RX_UUID       { 0x24, 0xdc, 0x0e, 0x6e, 0x03, 0x40, \
                                          0xca, 0x9e, 0xe5, 0xa9, 0xa3, 0x00, \
                                          0xb5, 0xf3, 0x93, 0xe0 }
#define CS_CHARACTERISTIC_TX_LONG_UUID  { 0x24, 0xdc, 0x0e, 0x6e, 0x04, 0x40, \
```

ble_custom.h

```
                                        0xca, 0x9e, 0xe5, 0xa9, 0xa3, 0x00, \
                                        0xb5, 0xf3, 0x93, 0xe0 }
#define CS_CHARACTERISTIC_RX_LONG_UUID  { 0x24, 0xdc, 0x0e, 0x6e, 0x05, 0x40, \
                                        0xca, 0x9e, 0xe5, 0xa9, 0xa3, 0x00, \
                                        0xb5, 0xf3, 0x93, 0xe0 }


#define ATT_DECL_CHAR() \
    { ATT_DECL_CHARACTERISTIC_128, PERM(RD, ENABLE), 0, 0 }
#define ATT_DECL_CHAR_UUID_16(uuid, perm, max_length) \
    { uuid, perm, max_length, PERM(RI, ENABLE) | PERM(UUID_LEN, UUID_16) }
#define ATT_DECL_CHAR_UUID_32(uuid, perm, max_length) \
    { uuid, perm, max_length, PERM(RI, ENABLE) | PERM(UUID_LEN, UUID_32) }
#define ATT_DECL_CHAR_UUID_128(uuid, perm, max_length) \
    { uuid, perm, max_length, PERM(RI, ENABLE) | PERM(UUID_LEN, UUID_128) }
#define ATT_DECL_CHAR_CCC()                                                 \
    { ATT_DESC_CLIENT_CHAR_CFG_128, PERM(RD, ENABLE) | PERM(WRITE_REQ, ENABLE), \
      0, PERM(RI, ENABLE) }
#define ATT_DECL_CHAR_USER_DESC(max_length)                         \
    { ATT_DESC_CHAR_USER_DESC_128, PERM(RD, ENABLE), max_length, \
      PERM(RI, ENABLE) }


enum cs_idx_att
{
    /* TX Characteristic */
    CS_IDX_TX_VALUE_CHAR,
    CS_IDX_TX_VALUE_VAL,
    CS_IDX_TX_VALUE_CCC,
    CS_IDX_TX_VALUE_USR_DSCP,
```

ble_custom.h

```
    /* RX Characteristic */
    CS_IDX_RX_VALUE_CHAR,
    CS_IDX_RX_VALUE_VAL,
    CS_IDX_RX_VALUE_CCC,
    CS_IDX_RX_VALUE_USR_DSCP,

    /* TX Long Characteristic */
    CS_IDX_TX_LONG_VALUE_CHAR,
    CS_IDX_TX_LONG_VALUE_VAL,
    CS_IDX_TX_LONG_VALUE_USR_DSCP,

    /* RX Long Characteristic */
    CS_IDX_RX_LONG_VALUE_CHAR,
    CS_IDX_RX_LONG_VALUE_VAL,
    CS_IDX_RX_LONG_VALUE_USR_DSCP,

    /* Max number of characteristics */
    CS_IDX_NB,
};

#define CS_TX_VALUE_MAX_LENGTH          20
#define CS_RX_VALUE_MAX_LENGTH          20
#define CS_USER_DESCRIPTION_MAX_LENGTH  16
#define CS_TX_LONG_VALUE_MAX_LENGTH     40
#define CS_RX_LONG_VALUE_MAX_LENGTH     40

#define CS_TX_CHARACTERISTIC_NAME       "TEMPERATURE_VALUE"
```

ble_custom.h

```
#define CS_RX_CHARACTERISTIC_NAME       "RX_VALUE"
#define CS_TX_LONG_CHARACTERISTIC_NAME  "TX_LONG_VALUE"
#define CS_RX_LONG_CHARACTERISTIC_NAME  "RX_LONG_VALUE"


/* List of message handlers that are used by the custom service
 * application manager */
#define CS_MESSAGE_HANDLER_LIST                                         \
    DEFINE_MESSAGE_HANDLER(GATTC_READ_REQ_IND, GATTC_ReadReqInd),   \
    DEFINE_MESSAGE_HANDLER(GATTC_WRITE_REQ_IND, GATTC_WriteReqInd), \
    DEFINE_MESSAGE_HANDLER(GATTM_ADD_SVC_RSP, GATTM_AddSvcRsp),     \
    DEFINE_MESSAGE_HANDLER(GATTC_CMP_EVT, GATTC_CmpEvt),            \
    DEFINE_MESSAGE_HANDLER(GATTC_ATT_INFO_REQ_IND, GATTC_AttInfoReqInd)


/* Define the available custom service states */
enum cs_state
{
    CS_INIT,
    CS_SERVICE_DISCOVERD,
    CS_ALL_ATTS_DISCOVERED,
    CS_STATE_MAX
};


/* ----------------------------------------------------------------------
 * Global variables and types
 * ----------------------------------------------------------------------*/


struct cs_env_tag
{
```

ble_custom.h

```
    /* The value of service handle in the database of attributes in the stack */
    uint16_t start_hdl;

    /* The value of TX characteristic value */
    uint8_t tx_value[CS_TX_VALUE_MAX_LENGTH];

    /* CCCD value of TX characteristic */
    uint16_t tx_cccd_value;

    /* A flag that indicates that TX value has been changed */
    bool tx_value_changed;

    /* A flag that indicates that PDU has been sent over the air */
    bool sent_success;

    /* The value of RX characteristic value */
    uint8_t rx_value[CS_RX_VALUE_MAX_LENGTH];

    /* CCCD value of RX characteristic */
    uint16_t rx_cccd_value;

    uint8_t tx_long_value[CS_TX_LONG_VALUE_MAX_LENGTH];

    uint8_t rx_long_value[CS_RX_LONG_VALUE_MAX_LENGTH];

    /* A flag that indicates that RX value has been changed, to be used by
     * application */
    bool rx_value_changed;
```

ble_custom.h

```
    bool rx_long_value_changed;
    /* The state machine for service discovery
     * (it is not used for server role) */
    uint8_t state;

    /* Custom service */
    uint16_t cnt_notifc;
    uint8_t val_notif;
};

extern struct cs_env_tag cs_env[];

extern void CustomService_Env_Initialize(void);

extern void CustomService_ServiceAdd(void);

extern int GATTM_AddSvcRsp(ke_msg_id_t const msgid,
                           struct gattm_add_svc_rsp const *param,
                           ke_task_id_t const dest_id,
                           ke_task_id_t const src_id);

extern int GATTC_ReadReqInd(ke_msg_id_t const msg_id,
                            struct gattc_read_req_ind const *param,
                            ke_task_id_t const dest_id,
                            ke_task_id_t const src_id);

extern int GATTC_WriteReqInd(ke_msg_id_t const msg_id,
```

ble_custom.h

```
                                struct gattc_write_req_ind const *param,
                                ke_task_id_t const dest_id,
                                ke_task_id_t const src_id);

extern void CustomService_SendNotification(uint8_t conidx, uint8_t attidx,
                                           uint8_t *value, uint8_t length);

extern int GATTC_CmpEvt(ke_msg_id_t const msg_id,
                        struct gattc_cmp_evt const *param,
                        ke_task_id_t const dest_id,
                        ke_task_id_t const src_id);

extern int GATTC_AttInfoReqInd(ke_msg_id_t const msg_id,
                               struct gattc_read_req_ind const *param,
                               ke_task_id_t const dest_id,
                               ke_task_id_t const src_id);

/* ----------------------------------------------------------------------------
 * Close the 'extern "C"' block
 * ---------------------------------------------------------------------------- */
#ifdef __cplusplus
}
#endif    /* ifdef __cplusplus */

#endif    /* BLE_CUSTOM_H */
```

Appendix B

RTS.m

```
clear


%   DEFINABLE VALUES


BLE_DEVICE_NAME = "Peripheral_Server";  %   BLE advertisement string


BLE_SERVICE_UUID = "E093F3B5-00A3-A9E5-9ECA-40016E0EDC24";
BLE_CHARACT_UUID = "E093F3B5-00A3-A9E5-9ECA-40026E0EDC24";


MEMORY_TIME = 10;


UPDATE_FREQUENCY = 100;


%   NON-DEFINABLE VALUES


t = 0;                                  % time
t_0 = 0;                                % time of measurements start


Vcc = 0;
T1 = 0;
T3 = 0;


t_pause = 1000 / UPDATE_FREQUENCY - 150


MAX_POINTS = round(UPDATE_FREQUENCY * MEMORY_TIME);
if(MAX_POINTS < 2)
```

```
    MAX_POINTS = 2;
end


BLE_IS_INITED = 0;


%   BLE CONNECTION INITIALIZATION


%       TRY TO SEARCH BLE DEVICE
while BLE_IS_INITED == 0
    RTS_INIT
end


%   LIVE PLOT INITIALIZATION


figure


hold on


% Create three animated lines for plotting
line_T1 = animatedline('Color', 'r', 'LineWidth', 3, ...
    'MaximumNumPoints', MAX_POINTS);
line_T3 = animatedline('Color', 'b', 'LineWidth', 3, ...
    'MaximumNumPoints', MAX_POINTS);
line_Vcc =animatedline('Color', 'g', 'LineWidth', 3, ...
    'MaximumNumPoints', MAX_POINTS);


legend('T_1 [C°]', 'T_2 [C°]', 'V_{cc} [V]')
```

```
grid on;


RTS_LOOKUP;      % Load lookup values


t_0 = posixtime(datetime('now'));


%   MAIN CYCLE
while 1
    try
        while 1
            %   VALUE UPDATE CYCLE


            %   Read the value from BLE device
            [Vcc, R_T1, R_T3, t] = RTS_READ(char, t_0);


            T1 = interp1(LOOKUP_R31, LOOKUP_TEMPERATURE, R_T1);
            T2 = Vcc;
            T3 = interp1(LOOKUP_R33, LOOKUP_TEMPERATURE, R_T3);


            %   Break the cycle if plot is closed
            if(ishandle(line_T1) == 0 || ...
                    ishandle(line_Vcc) == 0 || ...
                    ishandle(line_T3) == 0)
                clear all
                clc
                return;
            end
```

```
            %   Add new points to the plot
            addpoints(line_T1, t, T1)
            addpoints(line_Vcc, t, T2)
            addpoints(line_T3, t, T3)

            %   Scaling the plot time axis
            if(t < MEMORY_TIME)
                xlim([0 t]);
            else
                xlim([t-MEMORY_TIME t]);
            end

            %   Update the plot
            drawnow update

            if(t_pause > 0)
                % Java function is used for pause as more precise
                java.lang.Thread.sleep(t_pause);
            end
        end
    catch
        %   RECONNECTION ATTEMPT
        clear ble_device char;
        BLE_IS_INITED = 0;

        %   Break the cycle on plot closing
        if(ishandle(line_T1) == 0 || ...
                ishandle(line_Vcc) == 0 || ...
                ishandle(line_T3) == 0)
```

```
            clear all
            clc
            return;
        end

        while BLE_IS_INITED == 0
            RTS_INIT;
        end
    end
end
```

RTS_INIT.m

```
    %   SEARCHING THE RSL10 DEVICE
    if (height(blelist("Name", BLE_DEVICE_NAME)) == 0)
        warning("BLE device is not found!");
        BLE_IS_INITED = 0;
    else
        %   CONNECTING TO THE RSL10
        ble_device = ble(BLE_DEVICE_NAME);

        %   DEFINING CHARACTERISTIC
        char = characteristic(ble_device, BLE_SERVICE_UUID,
    BLE_CHARACT_UUID);

        BLE_IS_INITED = 1;
    end
```

RTS_LOOKUP.m

```
LOOKUP_TEMPERATURE = linspace(-55, 155, 43);
LOOKUP_R31 = [142 96.615 66.562 46.4 32.708 ...
              23.302 16.77 12.186 8.937 6.6125 ...
              4.9342 3.712 2.8145 2.15 1.6544 ...
              1.2819 1 0.78514 0.62031 0.49304 ...
              0.39417 0.3169 0.25616 0.208 0.17 ...
              0.13952 0.11505 0.095302 0.079296 0.066263 ...
              0.055 0.046 0.039 0.033 0.028 ...
              0.024 0.021 0.0181 0.0156 0.0135 ...
              0.0117 0.0102 0.00896];

LOOKUP_R33 = [96.158 66.892 47.127 33.606 24.243 ...
              17.681 13.032 9.702 7.2923 5.5314 ...
              4.2325 3.2657 2.54 1.9907 1.5716 ...
              1.2494 1 0.8055 0.65288 0.53229 ...
              0.43654 0.2487 0.29819 0.24837 0.20787 ...
              0.17479 0.14763 0.12523 0.10667 0.091227 ...
              0.078319 0.067488 0.058363 0.050647 0.044098 ...
              0.03852 0.033752 0.029663 0.026146 0.023111 ...
              0.020484 0.018203 0.018202];
```

RTS_READ.m

```
function [v_batt, R_T1, R_T3, t] = RTS_READ(char, starttime)

%   UPDATE THE CURRENT TIME FOR PLOTTING
t = posixtime(datetime('now')) - starttime;

%   UPDATE CHARACTERISTIC'S VALUE
line = read(char);
```

```
%   CALCULATE BATTERY VOLTAGE
temp = line(1:4);
temp = typecast(uint8(temp), 'uint32');
v_batt = double(swapbytes(temp));
v_batt = 2 * v_batt / 16384;


%   CALCULATE T_1 THERMISTOR RESISTANCE
temp = line(5:8);
temp = typecast(uint8(temp), 'uint32');
v_r31 = double(swapbytes(temp));
v_r31 = 2 * v_r31 / 16384;
r_31 = 10^6;
i_1 = (v_batt - v_r31) / r_31;
R_T1 = v_r31 / i_1;
R_T1 = R_T1 / 10^6;


%   CALCULATE T_3 THERMISTOR RESISTANCE
temp = line(9:12);
temp = typecast(uint8(temp), 'uint32');
v_r33 = double(swapbytes(temp));
v_r33 = 2 * v_r33 / 16384;
r_33 = 10^6;
i_3 = (v_batt - v_r33) / r_33;
R_T3 = v_r33 / i_3;
R_T3 = R_T3 / 10^6;


end
```